TUGAS AKHIR – IF184802

DETEKSI API BERBASIS DATA VIDEO MENGGUNAKAN METODE OPTICAL FLOW DAN SUPPORT VECTOR MACHINE

SIRRIA PANAH ALAM
NRP 05111540000017

Dosen Pembimbing I
Dr.Eng Chastine Fatichah, S.Kom., M.Kom.

Dosen Pembimbing II
Dini Adni Navastara, S.Kom., M.Sc.

Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

TUGAS AKHIR – IF184802

DETEKSI API BERBASIS DATA VIDEO MENGGUNAKAN METODE OPTICAL FLOW DAN SUPPORT VECTOR MACHINE

SIRRIA PANAH ALAM
NRP 05111540000017

Dosen Pembimbing I
Dr.Eng Chastine Fatichah,S.Kom., M. Kom.

Dosen Pembimbing II
Dini Adni Navastara, S.Kom., M.Sc.

Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESIS – IF184802

FIRE DETECTION BASED ON VIDEO DATA USING OPTICAL FLOW AND SUPPORT VECTOR MACHINE

SIRRIA PANAH ALAM
NRP 05111540000017

First Advisor
Dr.Eng Chastine Fatichah, S.Kom., M. Kom.

Second Advisor
Dini Adni Navastara, S.Kom., M.Sc.

Department of Informatics
Faculty of Information and Communication Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2019

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

DETEKSI API BERBASIS DATA VIDEO MENGGUNAKAN METODE OPTICAL FLOW DAN SUPPORT VECTOR MACHINE

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Cerdas dan Visi
Program Studi S-1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:
SIRRIA PANAH ALAM
NRP: 05111540000017

Disetujui oleh Pembimbing Tugas Akhir:

1. Dr.Eng Chastine Fatichah, S.Kom., M.Kom. .....
   (NIP 19751220 200112 2 002) (Pembimbing 1)

2. Dini Adni Navastara, S.Kom., M.Sc. .....
   (NIP. 19851017 201504 2 001) (Pembimbing 2)

SURABAYA
Januari, 2019

(Halaman ini sengaja dikosongkan)

DETEKSI API BERBASIS DATA VIDEO MENGGUNAKAN METODE OPTICAL FLOW DAN SUPPORT VECTOR MACHINE

Nama Mahasiswa	**: Sirria Panah Alam**
NRP	**: 05111540000017**
Jurusan	**: Informatika, FTIK-ITS**
Dosen Pembimbing 1	**: Dr.Eng Chastine Fatichah, S.Kom., M.Kom.**
Dosen Pembimbing 2	**: Dini Adni Navastara, S.Kom., M.Sc.**

ABSTRAK

Kebakaran hutan dan lahan di Indonesia telah menjadi krisis lingkungan tahunan. Tercatat 2,6 juta hektar lahan di Indonesia terbakar pada bulan Juni hingga Oktober tahun 2015. Kebakaran menimbulkan kerusakan lingkungan dan banyak kerugian finansial utamanya apabila terjadi di pemukiman masyarakat. Maka dari itu diperlukan suatu pendeteksi adanya api untuk mendeteksi kebakaran lebih awal. Penggunaan data video untuk mendeteksi adanya kebakaran dilakukan dengan mengekstraksi berbagai karakteristik api, berupa tekstur dan pola gerak api. Maka dari itu dalam penelitian ini dilakukan pendeteksian api pada data video menggunakan ekstraksi fitur tekstur Local Binary Pattern (LBP) serta ekstraksi gerak menggunakan metode Optical Flow.

Pada tugas akhir ini dilakukan segmentasi terlebih dahulu pada setiap frame video. Dari hasil segmentasi diperoleh citra potongan kandidat area api. Citra tersebut kemudian diekstraksi fitur teksturnya menggunakan LBP dan frame pada citra tersebut di ektraksi fitur geraknya menggunakan metode optical flow. Selanjutnya, kedua fitur tersebut diklasifikasi menggunakan metode Support Vector Machine (SVM). Model dievaluasi dengan menggunakan stratified k-fold untuk dipisahkan menjadi dua subset yaitu data proses pembelajaran dan data validasi/evaluasi

dengan jumlah k=10. Nilai akurasi terbaik diperoleh dengan ekstraksi fitur LBP dan Optical Flow Lucas Kanade menggunakan metode SVM dengan kernel linear yaitu sebesar 95.96%, serta menghasilkan presisi sebesar 93.76% dan recall sebesar 99.73%

Kata kunci: *Optical Flow, Local Binary Pattern, Deteksi Api, Data Video, SVM.*

FIRE DETECTION BASED ON VIDEO DATA USING OPTICAL FLOW AND SUPPORT VECTOR MACHINE

Student's Name : Sirria Panah Alam
Student's ID : 05111440000017
Department : Informatics, Faculty of ICT-ITS
First Advisor : Dr.Eng Chastine Fatichah, S.Kom., M.Kom.
Second Advisor : Dini Adni Navastara, S.Kom., M.Sc.

ABSTRACT

Forest and land fires have become annual environmental crises in Indonesia. Around 2.6 million hectares were burned in June to October 2015. Therefore, an early warning system is needed to detect the presence of fire. Video data can be used to prove the existence of a fire carried out by extracting its various characteristics, consisting of the texture and pattern of fire movements. This study focuses on fire detection using video data through textures extraction of Local Binary Pattern (LBP) and motion extraction using the Optical Flow method.

First, we perform a segmentation on each video frames. From this process, we obtained the image of the fire area. It will be extracted using LBP and optical flow method to obtain the texture and movement features of the fire. The features are classified using the Support Vector Machine (SVM) method. The model is evaluated using stratified k-fold to be separated into two subsets, learning process data and validation/ evaluation data with 10 number of k-folds. The best result of true positive and true negative obtained from classification using Local Binary Pattern and Lucas Kanade Optical Flow feature extraction with SVM method using linear kernel. The value of accuration is 95.96%, with precission 93.76% and recall 99.73%

Keywords: *Optical Flow, Local Binary Pattern, Fire Detection, Video Data, SVM*

KATA PENGANTAR

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji bagi Allah SWT yang telah melimpahkan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul:

"**DETEKSI API BERBASIS DATA VIDEO MENGGUNAKAN METODE OPTICAL FLOW DAN SUPPORT VECTOR MACHINE**"

Terselesaikannya Tugas Akhir ini tidak terlepas dari bantuan dan dukungan banyak pihak, Oleh karena itu melalui lembar ini penulis ingin mengucapkan terima kasih dan penghormatan kepada:

1. Allah SWT serta junjungan Nabi Muhammad SAW, karena limpahan rahmat dan karunia-Nya penulis dapat menyelesaikan Tugas Akhir dan juga perkuliahan di Informatika ITS.
2. Kedua orangtua penulis, Batur Subagyo dan Hetty Suprihatin dan anggota keluarga lainnya terutama kedua kakak saya Alfi Revolusi dan Senja Sakti yang tiada hentinya memberikan dukungan doa, moral, dan material kepada penulis sehingga penulis dapat menyelesaikan Tugas Akhir ini.
3. Dr.Eng Chastine Fatichah, S.Kom., M.Kom. dan Dini Adni Navastara, S.Kom., M.Sc. selaku pembimbing I dan II yang telah membimbing dan memberikan motivasi, nasihat dan bimbingan dalam menyelesaikan Tugas Akhir ini.

4. Dr. Eng. Darlis Herumurti, S.Kom., M.Kom. selaku Ketua Departemen Informatika ITS dan segenap dosen dan karyawan Departemen Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa studi di Informatika ITS.
5. Kakak tingkat saya Mas Hamdi dan Mas Adit yang rela meluangkan waktunya untuk berdiskusi dalam pengerjaan tugas akhir ini.
6. Rekan saya Ronald Andrean dan Hendry Wiranto yang bersedia memberikan waktu untuk berdiskusi dalam pengerjaan Tugas Akhir.
7. Sahabat saya Agatha Putri, Syavira Tiara, Astrid Febrianca, Mutia Rahmi, Fatimatuz Zulfa, Aulia Teaku, Nafingatun Ngaliah, Dara Tursina, Afrian Muftichatus sebagai penyemangat dan tempat berbagi cerita bagi penulis
8. Teman-teman yang ada di dalam Laboratorium Komputasi Cerdas & Visi yang memberikan kesempatan penulis untuk fokus mengerjakan Tugas Akhir ini di ruangan tersebut.
9. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa laporan Tugas Akhir ini masih memiliki banyak kekurangan. Oleh karena itu dengan segala kerendahan hati penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan penulis kedapannya. Selain itu, penulis berharap laporan Tugas Akhir ini dapat berguna bagi pembaca secara umum.

Surabaya, Januari 2019

DAFTAR ISI

DAFTAR TABEL

DAFTAR KODE SUMBER

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kebakaran hutan dan lahan di Indonesia telah menjadi krisis lingkungan tahunan. Tercatat 2,6 juta hektar lahan di Indonesia terbakar pada bulan Juni hingga Oktober tahun 2015. [1] Kebakaran menimbulkan kerusakan lingkungan dan banyak kerugian finansial utamanya apabila terjadi di pemukiman masyarakat. Maka dari itu diperlukan suatu pendeteksi adanya api untuk mendeteksi kebakaran lebih awal. Sistem alarm yang telah ada saat ini menggunakan sensor optik, panas dan ion yang dapat bekerja dengan baik dan efisien namun hanya bisa mendeteksi adanya kebakaran dalam ruang tertutup dan jangkauan yang terbatas.

Penelitian sebelumnya yang menjadi referensi utama tugas akhir ini adalah tugas akhir oleh Hamdi A.M pada tahun 2016 [2] melakukan riset pada studi kasus yang sama. Pada tugas akhir sebelumnya tidak menerapkan deteksi tekstur. Hanya diterapkan deteksi warna berdasarkan channel *RGB* dan deteksi gerak menggunakan *Gaussian Mixture Model*. Selain itu dalam ekstraksi fitur dilakukan dengan metode *wavelet*. Berdasarkan penelitian oleh Santana penelitian tersebut dapat dikembangkan dengan mengganti channel warna menggunakan *HSI/HSV*. Menurut Wang tahun 2016 [3], serta Avegerinakis tahun 2017 [4], deteksi api memperoleh hasil optimal dengan menggunakan deteksi tekstur *Local Binary Pattern*, serta deteksi gerak menggunakan *Optical flow* yang dijadikan fitur sebelum dilakukan klasifikasi menggunakan *Support Vector Machine.*

Oleh karena itu, dalam tugas akhir ini diimplementasikan perangkat lunak pendeteksi api menggunakan teknologi visi komputer berdasarkan deteksi warna menggunakan probabilitas warna dan segmentasi menggunakan *Hue*, *Saturation*, *Intensity* serta deteksi gerak menggunakan *Optical Flow*. Ekstraksi fitur tekstur menggunakan metode Local Binary Pattern, dan klasifikasi piksel menggunakan *Support Vector Machine*. Hasil klasifikasi

digunakan sebagai penentuan adanya api. Hasil dari tugas akhir ini diharapkan dapat memberikan penjelasan mengenai metode pendeteksian api berbasis sensor visual yang memanfaatkan berbagai karakteristik dari api.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana melakukan deteksi warna api pada piksel frame video?
2. Bagaimana melakukan deteksi tekstur api pada piksel frame video?
3. Bagaimana mendeteksi gerak api pada setiap frame video?
4. Bagaimana mengklasifikasikan fitur-fitur tersebut sebagai penentuan adanya api?

1.3 Batasan Permasalahan

Berikut beberapa hal yang menjadi batasan masalah dalam pengerjaan tugas akhir:

1. Implementasi menggunakan bahasa pemrograman Python.
2. Jumlah piksel objek api yang dideteksi tidak kurang dari 5x5 piksel.
3. Data yang digunakan adalah video dengan panjang video 6-21 detik dengan ekstensi mp4.
4. Data Video memiliki ukuran 854 x 480 piksel, dengan channel *Red, Green, Blue* (RGB).
5. Warna api yang didefinisikan adalah *range* warna kuning hingga merah.
6. Kamera cenderung statis.
7. Pantulan objek api, termasuk kedalam objek api.
8. Data video berasal dari KMU Fire and Smoke Database, video *open source*, MIVIA *fire dataset*, dan video dari Youtube.

1.4 Tujuan

Tujuan dari tugas akhir ini adalah merancang dan membangun perangkat lunak pendeteksi api menggunakan data video secara real-time menggunakan metode *Optical Flow* dan *Support Vector Machine*.

1.5 Manfaat

Dengan adanya perangkat lunak ini diharapkan dapat membantu deteksi adanya kebakaran lebih awal dan lebih efisien utamanya untuk lingkungan yang terbuka.

1.6 Metodologi

Pembuatan Tugas Akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

1.6.1 Penyusunan Proposal Tugas Akhir

Proposal tugas akhir ini akan mendeskripsikan dan membahas mengenai rencana pembuatan aplikasi deteksi api menggunakan sensor visual. Secara detil, proposal tugas akhir ini berisi tentang beberapa bagian yaitu latar belakang diajukannya tugas akhir, rumusan masalah yang diangkat, batasan masalah untuk tugas akhir, tujuan dari pembuatan tugas akhir, dan manfaat dari hasil pembuatan tugas akhir. Selain itu, dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan tugas akhir dan ringkasan isi yang membahas metode yang akan digunakan dalam tugas akhir. Sub bab metodologi merupakan penjelasan mengenai tahapan penyusunan tugas akhir. Terdapat pula sub bab jadwal pengerjaan yang menjelaskan jadwal pengerjaan tugas akhir dan di akhir bagian terdapat daftar pustaka untuk mencantumkan referensi yang digunakan dalam tugas akhir

1.6.2 Studi Literatur

Pada tahap ini dilakukan pencarian, pengumpulan, penyaringan, pemahaman, dan pembelajaran literature yang berhubungan dengan reduksi *size frame*, deteksi gerak

menggunakan *optical flow*, segmentasi dan deteksi warna piksel menggunakan *hue, saturation, intensity* serta metode klasifikasi *support vector machines*. Literatur yang digunakan meliputi: buku, referensi, jurnal, dan dokumentasi internet.

1.6.3 Analisis dan Desain Perangkat Lunak

Tahap ini meliputi perancangan sistem berdasarkan studi literatur dan pembelajaran konsep teknologi dari perangkat lunak yang ada. Tahap ini mendefinisikan alur dari implementasi. Langkah-langkah yang dikerjakan juga didefinisikan pada tahap ini. Pada tahapan ini dilakukan desain sistem dan desain proses-proses yang ada.

1.6.4 Implementasi Perangkat Lunak

Pada tahap ini dilakukan pembangunan perangkat lunak sesuai dengan rancangan yang dibangun menggunakan bahasa pemrograman Python digunakan IDE yaitu Spyder dan menggunakan library openCV

1.6.5 Pengujian dan Evaluasi

Tahap pengujian dan evaluasi pada perangkat lunak yang telah dibuat bertujuan untuk mengetahui kemampuan algoritma yang dipakai, mengamati kinerja sistem, serta mengidentifikasi kendala yang mungkin timbul. Parameter yang diujikan adalah parameter *threshold* pada deteksi warna piksel, dan nilai konstanta C, kernel pada klasifikasi *support vector machines, size frame* yang diproses, dan penggunaan tahap perhitungan luasan region.

Pada tahapan ini dilakukan uji coba terhadap aplikasi yang telah dibuat. Pengujian ini betujuan untuk mengetahui unjuk kerja dari aplikasi pengenalan aktivitas pada data video. Pada uji coba ini, langkah pertama adalah melatih sistem dengan data training untuk membentuk model pendeteksi api. Langkah selanjutnya akan diberikan data uji untuk menguji model aktivitas yang telah

terbentuk serta menghasilkan prediksi ada tidaknya api dari data uji. Parameter yang akan digunakan sebagai bahan uji coba adalah *kernel SVM*, proses pembagian data dan jenis *optical flow*

1.6.6 Penyusunan Buku

Pada tahap ini dilakukan penyusunan buku yang menjelaskan seluruh konsep, teori dasar dari metode yang digunakan, implementasi, serta hasil yang telah dikerjakan sebagai dokumentasi dari pelaksanaan Tugas Akhir.

1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan Tugas Akhir adalah sebagai berikut:

Bab I Pendahuluan
Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan Tugas Akhir.

Bab II Tinjauan Pustaka
Bab ini berisi kajian teori dari metode dan algoritma yang digunakan dalam penyusunan Tugas Akhir ini. Secara garis besar, bab ini berisi tentang segmentasi *hue, saturation,value ,local binary pattern, optical flow* yang digunakan

Bab III Perancangan Sistem
Bab ini berisi pembahasan mengenai perancangan dari metode *local binary pattern dan optical flow* yang digunakan untuk pendeteksiapi pada data video.

Bab IV Implementasi
Bab ini membahas implementasi dari perancangan yang telah dibuat pada bab sebelumnya. Penjelasan berupa kode yang digunakan untuk proses implementasi.

Bab V Uji Coba Dan Evaluasi

Bab ini membahas tahapan uji coba, kemudian hasil uji coba dievaluasi terhadap kinerja dari sistem yang dibangun.

Bab VI Kesimpulan dan Saran

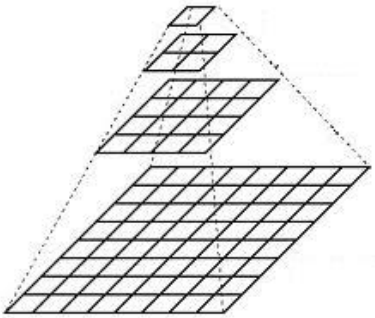
Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses dan tertulis saat pengerjaan Tugas Akhir, dan saran untuk pengembangan solusi ke depannya

BAB II
TINJAUAN PUSTAKA

Bab ini berisi pembahasan mengenai teori-teori dasar yang digunakan dalam Tugas Akhir. Teori-teori tersebut diantaranya adalah *Local Binary Pattern* dan *Optical Flow* serta beberapa teori lain yang mendukung pembuatan Tugas Akhir. Penjelesan ini bertujuan untuk memberikan gambaran secara umum dan berguna sebagai penunjang sistem yang dibangun.

2.1 Gaussian Pyramid

Gaussian pyramid digunakan dalam melakukan reduksi resolusi citra "Image pyramid" adalah kumpulan citra dengan resolusi yang semakin menurun, yang disusun dalam bentuk pyramid. Citra yang tereduksi resolusinya akan berkurang menjadi seperempat dari resolusi awal. Hal ini dilakukan untuk mempercepat proses perhitungan. Ilustrasi gaussian pyramid dapat dilihat pada Gambar 2.1



Gambar 2.1 Ilustrasi Gaussian Pyramid 4 level [5]

Gaussian pyramid dilakukan dengan dua operasi, yaitu *smoothing* dan *down sampling*. [5] *Smoothing* dilakukan dengan menggunakan filter 5x5. *Smoothing* dilakukan dengan menggunakan Persamaan (2.1).

$$g_1 = w * g_0 \tag{2.1}$$

Dimana w adalah filter 5x5. Detail persamaan setiap piksel dilakukan menggunakan Persamaan (8.2) dimana g_1 adalah citra hasil smoothing dari citra asli g_0.

$$g_1(i,j) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n).g_0\,(i+m, j+n) \tag{2.2}$$

Setelah mendapatkan citra yang telah di lakukan *smoothing*, langkah selanjutnya adalah melakukan *down sampling*. *Down sampling* dilakukan untuk mengubah resolusi citra asli menjadi resolusi yang lebih kecil. *Down sampling* dilakukan dengan Persamaan (8.3) pada persamaan g_2 ini mewakili citra hasil *down sampling* dari citra hasil *smoothing* g_1 sebelumnya.

$$g_2(i,j) = g_1\,(2i, 2j) \tag{2.3}$$

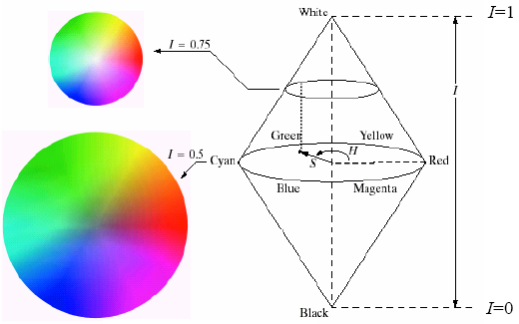
Untuk melakukan reduksi, dilakukan menggunakan dua proses tersebut, yaitu *smoothing* dan *down sampling*. Perhitungan dapat dilakukan dengan menggabungkan ke dua persamaan tersebut menjadi Persamaan (2.4), yang mana g_1 adalah citra hasil gabungan *smoothing* dan *down sampling* dari citra asli g_0.

$$g_1(i,j) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n).g_0\,(2i+m, 2\,j+n) \tag{2.4}$$

2.2 Segmentasi Hue, Saturation, Intensity

Hue, *saturation*, *intensity* adalah sebuah model warna yang dapat dideskripsikan oleh mata manusia. *Hue* menyatakan warna asli dari suatu piksel, *saturation* merepresentasikan besar derajat warna tersebut terpengaruh oleh cahaya, dan *value* atau *intensity* mendeskripsikan level

keabuan dari suatu warna. [6] *Hue* dan *saturation* memberikan informasi *chromatic* dari warna, sedangkan intensity memberikan informasi *achromatic*. Dalam model warna, *Hue* adalah sudut besarnya 0^0 hingga 2π. Saturation adalah jarak ke sumbu vertical besarnya 0 hingga 1, sedangkan *intensity/value* adalah ketinggian di sepanjang sumbu vertikal yang besanya 0 hingga 1. Di ilustrasikan pada Gambar 2.2 berikut.



Gambar 2.2 Ilustrasi Model Warna HIS/HSV [6]

Dengan menggunakan repersentasi HSI yang membawa informasi warna dari sutau gambar, dapat dilakukan suatu segmentasi gambar berdasarkan warna. Proses segmentasi diawali dengan mengubah citra RGB menjadi citra HSI/HSV. Jika diberikan sebuah gambar dengan format warna RGB, maka nilai dari komponen *Hue* atau H dari piksel RGB tersebut didapatkan berdasarkan Persamaan (2.5).

$$H = \begin{cases} \theta & if\ B \leq G \\ 360 - \theta & if\ B > G \end{cases} \tag{2.5}$$

Dengan Θ sesuai dengan Persamaan (2.6)

$$\theta = \cos -1 \left\{ \frac{\frac{1}{2}[(R-G) + (R-B)]}{[(R-G)^2 + (R-B)(G-B)]^{\frac{1}{2}}} \right\} \quad (2.6)$$

Besarnya Komponen Saturasi (S) adalah sesuai dengan persamaan (2.7)

$$S = 1 - \frac{3}{(R+G+B)} [\min(R,G,B)] \quad (2.7)$$

Besarnya komponen *Intensity*/*value* (I) ditetapkan berdasarkan Persamaan (2.8)

$$I = \frac{1}{3} (R+G+B) \quad (2.8)$$

Diasumsikan bahwa nilai RGB telah dinormalisasi dalam range [0,1], dan besanya Ɵ diukur dengan memperhatikan sumbu merah pada HIS sebagaimana Gambar 2.2. komponen hue dapat dinormalisasi dalam range [0,1] dengan membagi cara dibagi 360^0. Sehingga semua komponen berada pada range [0,1].

2.3 Local Binary Pattern

Local Binary Pattern (LBP) merupakan metode ekstraksi fitur yang digunakan untuk mendeteksi tekstur. LBP menggunakan dua pendekatan yaitu pendekatan struktur dan pendekatan statistika [7]. Operasi LBP bekerja pada blok piksel 3x3 dari sebuah citra. Piksel-piksel pada blok tersebut kemudian diberikan *threshold* oleh piksel tengah, lalu dikalikan kuadrat dua, dan kemudian dijumlahkan untuk mendapatkan label baru untuk piksel tengah. Karena sebuah ketetanggaan dari piksel yang terdiri dari 8 piksel, sejumlah 28 = 256 label yang berbeda yang mungkin didapatkan bergantung pada nilai keabuan relative dari piksel tengah pada ketetanggaan piksel. Nilai desimal dari 8bit dari (*LBP code*) dapat dinyatakan dalam persamaan (2.9).

$$LBP_{P,R}\,(x_c, y_c) = \sum_{p=0}^{p-1} s(g_p - g_c)2^p \tag{2.9}$$

2.4 Optical Flow

Optical flow digunakan untuk menghitung pergerakan piksel dari citra yang berurutan. [8] *Optical flow* menerapkan kerapatan (*point to point*) piksel yang bersesuaian. Kesulitan dalam metode ini adalah menentukan dimana piksel pada suatu citra pada waktu *t* dan pada waktu *t+1*.

Optical flow bekerja dalam asumsi intensitas piksel pada suatu objek tidak berubah, antar satu frame dengan frame berikutnya dan piksel-piksel yang berketetanggaan memiliki gerakan yang serupa.

Dimisalkan terdapat suatu piksel $I(x,y,t)$ pada frame pertama, bergerak dengan jarak (d_x,d_y) pada frame berikutnya yang diambil setelah d_t time. Maka apabila piksel tersebut sama dan intensitas nya tidak berubah maka diperoleh Persamaan (2.10)

$$I\,(x, y, t) = I(x + dx, y + dy, t + dt) \tag{2.10}$$

Kemudian gunakan aproksimasi Taylor Series pada Persamaan (2.11)

$$f_x u + f_y u + f_t = 0 \tag{2.11}$$

Dimana, digunakan Persamaan (2.12) dalam penyelesaiannya.

$$f_x = \frac{\partial f}{\partial x}\ ;\ f_y = \frac{\partial f}{\partial x}$$

$$u = \frac{dx}{dt}\ ; v = \frac{dy}{dt} \tag{2.12}$$

Variabel f_x dan f_y adalah gradien dari citra, sedangkan f_t adalah gradien berdasarkan waktu. Tetapi (*u*,*v*) tidak diketahui. Kita tidak dapat menyelesaikan persamaan ini tanpa mengetahui dua variabel tersebut. Sehingga beberapa metode diperkenalkan untuk memecahkan persamaan tersebut, yaitu *Lucas Kanade* yang bekerja dengan baik pada teknik pencocokan titik yang mengobservasi titik-titik tertentu atau titik yang unik, ada pula *optical flow* yang megamati seluruh titik di semua area yang diobservasi menggunakan metode *Farneback.*

2.4.1 Metode Lucas Kanade

Pada asumsi sebelumnya, diketahui bahwa semua piksel yang berketetanggaan memiliki gerakan yang mirip. Metode Lucas- Kanade menggunakan blok piksel 3x3, sehingga 9 titik yang berketanggaan memiliki gerakan yang sama. Sehingga diperoleh Persamaan (2.13).

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum_i {f_{xi}}^2 & \sum_i f_{xi} f_y \\ \sum_i f_{xi} f_y & \sum_i {f_{yi}}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i f_{xi} f_{ti} \\ -\sum_i f_{yi} f_{ti} \end{bmatrix} \qquad (2.13)$$

Berdasarkan Persamaan (2.13) Persaman sebelumnya dapat diselesaikan.

2.4.2 Metode Farneback

Metode Farneback adalah metode yang memperkirakan pergerakan 2 frame yang berurutan menggunakan algoritma penjabaran *polynomial*, dimana ketetanggaan dari setiap piksel ditentukan sebagai *polynomial*. Pada penjelasan ini hanya dijelaskan *polynomial* kuadrat. [9]. Bentuk *polynomial* tersebut adalah.

$$f(x) = x^T A_1 x + b_1{}^T x + C_1 \tag{2.14}$$

Pada persamaan vector x merupakan vector berukuran 1x2 yang berisi nilai x, y yang merupakan nilai piksel yang diobservasi. A_1 adalah matriks berukuran 2x2 yang belum diketahui isinya, yang menyimpan informasi mengenai frame genap, b1 merupakan vektor berukuran 2x1 yang belum diketaui dan c1 adalah nilai scalar yang belum diketahui. Kemudian dicari fungsi baru f_2 dengan perpindahan sebesar d sehingga diperoleh persamaan (2.15)

$$\begin{aligned} f2(x) = f1(x-d) &= (x-d)^T A_1 (x-d) + b_1{}^T (x-d) + c \\ &= x^T A_1 x + (b_1 - 2A_1 d)^T x + d^T A_1 d - b_1^T d + c_1 \\ &= x^T A_2 x + b_2^T x + c_2 \end{aligned} \tag{2.15}$$

Dengan asumsi bahwa inteansitas piksel dari kedua frame yang berurutan tidak berubah, maka dapat dicari nilai koefisiennya

$$A_2 = A_1 \tag{2.16}$$

$$b_2 = b_1 - 2A_1 d \tag{2.17}$$

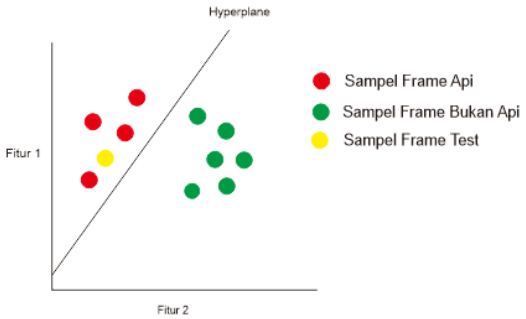
Dari persamaan (2.17) dapat dihitung nilai perpindahan optical flow dengan persamaan (2.18) dan (2.19)

$$2A_1 d = -(b_2 - b_1) \tag{2.18}$$

$$d = {}^1\!/_2 \, A_1^{-1}(b_2 - b_1) \tag{2.19}$$

2.5 Support Vector Machine

Support Vector Machine adalah metode klasifikasi yang mengklasifikasikan dua kelas, yaitu kelas +1 dan -1. Pada metode klasifikasi *support vector machines*, dibentuk suatu *hyperplane*. *Hyperplane* adalah garis pemisah yang memisahkan dua kelas yang berbeda. Dalam metode ini dikenal dengan istilah margin. Margin adalah jarak antara kelas +1 dengan kelas -1 yang paling dekat, selanjutnya dicari margin terpanjang antara dua kelas tersebut. Ilustrasi dari metode *support vector machines* dapat dilihat pada Gambar 2.3.



Gambar 2.3 Ilustrasi Support Vector Machine

Diberikan masukan berupa data belajar ($x1$,$x2$,$x3$,...,xn) dan masing-masing kelas dianotasikan $yi \in \{-1,+1\}$ untuk i = 1,2,3,…,l , dimana l adalah banyaknya data. Fungsi *hyperplane* dibuat dengan Persamaan (2.20). Dalam mencari nilai *w* dan *b* yang optimal, dilakukan dengan Persamaan (2.21).

$$w.x + b = 0 \tag{2.20}$$

$$\min \frac{1}{2} ||w||^2 + C \sum_{i=1}^{l} ti \tag{2.21}$$

Persamaan diatas mempunyai variabel C, dimana variabel tersebut adalah konstanta nilai pinalti dari kesalahan klasifikasi. Pencarian nilai w dan b dilakukan dengan batasan yang ditulis menggnakan Persamaan (2.22).

.

$$y_i \, (wx_i + b) + t_i \geq 1 \tag{2.22}$$

Persamaan (2.21) dilakukan untuk mencari nilai w dan b yang optimum. Fungsi tujuan Persamaan (2.21) berbentuk kuadrat. Untuk menyelesaikannya, bentuk tersebut ditransformasi kedalam bentuk dual space. Persamaan dual space dapat ditulis menggunakan Persamaan (2.23).

$$\max \sum_{i=1}^{l} a_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j a_i a_j {x_i}^T \, x_j \tag{2.23}$$

Dengan batasan pada Persamaan (2.24).

$$a_i \geq \sum_{i=0}^{l} a_i y_i = 0 \tag{2.24}$$

Untuk mencari nilai αi, digunakan *quadratic programming*. Setelah mendapatkan nilai αi, persamaan *hyperplane* dilakukan dengan Persamaan (8.19).

$$f = w^T z + b = \sum_{i=1}^{l} a_i y_i {x_i}^T z + b \tag{2.25}$$

Dimana *z* adalah data masukan. Pada banyak kasus, data yang diklasifikasikan tidak bisa langsung dipisahkan dengan garis yang linear. Oleh karena itu, digunakan metode kernel untuk mengatasi permasalahan tersebut. Dengan metode kernel, suatu data x di input space dimapping ke fitur space *F* dengan dimensi yang lebih tinggi. Salah satu kernel yang biasa dipakai adalah kernel RBF dan Polinomial. Persamaan kernel RBF dan Polinomial berurutan dapat dilihat pada Persamaan (2.26) dan Persamaan (2.27).

$$k\,(x,y) = \exp(-y|x-y|^2) \tag{2.26}$$

$$k\,(x,y) = (y(x^T\,y\,) + r)^p \tag{2.27}$$

Penggunaan fungsi kernel mengubah persamaan training. Persamaan dapat dilihat pada Persamaan (2.28).

$$\max \sum_{i=1}^{l} a_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j a_i a_j k(x_i, x_j) \tag{2.28}$$

Persamaan *hyperplane* diubah menjadi Persamaan (8.24).

$$f = \sum_{i,j=1}^{l} a_i y_i k(x_i, z) + b \tag{2.29}$$

2.6 Euclidean Distance

Euclidean Distance adalah metriks yang paling sering digunakan untuk menghitung kesamaan dua vektor. [10] Rumus *Euclidean Distance* adalah akar dari kuadrat perbedaan 2 vektor

(*root of square differences between 2 vectors*) sesuai dengan Persamaan (2.30)

$$d_{i,j} = \sqrt{\sum_{k=1}^{n} (x_{i,k} - X_{j,k})^2} \quad (2.30)$$

Dimana $d_{i,j}$ merupakan tingkat perbedaan (*dissimilarity degree*), n adalah jumlah vector, $x_{i,k}$ adalah poin *optical flow frame* sebelum dan $x_{j,k}$ adalah poin *optical frame* setelah bergerak.

2.7 Akurasi, Precision & Recall

Ketika membangun sebuah model klasifikasi, pertanyaan yang muncul adalah bagaimana mengetahui seberapa baik model tersebut. Mengevaluasi model klasifikasi dilakukan dengan mencaritahu seberapa baik hasil prediksi dari model tersebut. Penjelasan variabel ada pada *confusion matrix* pada Tabel 2.1

Tabel 2.1 *Confusion matrix*

		Kelas Prediksi	
		Api	Bukan Api
Kelas sebenarnya	Api	*True Positive* (TP)	*False Negative* (FN)
	Bukan Api	*False Positive* (FP)	*True Negative* (TN)

Keterangan :

1. *True Positive* (TP) pada video api dan dikenali oleh keluaran program sebagai video api.
2. *True Negative* (TN) pada video bukan api yang terekam, pada keluaran program tidak dikenali sebagai api.
3. *False Negative* (FN) pada video api, pada keluaran program tidak dikenali sebagai api.
4. *False Positive* (FP) pada video bukan api yang terekam, pada keluaran program dikenali sebagai api.

$$Recall = TP / (TP + FN) \quad (2.1)$$

$$Precision = TP / (TP + FP) \quad (2.2)$$

$$Akurasi = (TP + TN) / (TP + FP + TN + FN) \quad (2.3)$$

2.8 Python

Python adalah bahasa pemograman yang popular. *Python* sering dimanfaatkan dalam pengembangan web, perangkat lunak, penelitian, dan *system scripting*. Python dapat digunakan untuk menangani data besar dan melakukan operasi matematika yang kompleks. Python bekerja di berbagai *platform* seperti Windows, Mac, Linux, Raspberry Pi, dan lain-lain. Python dirancang untuk mudah dibaca, yaitu memiliki sintaks yang sederhana dan menggunakan bahasa inggris. [11]

2.9 OpenCV

OpenCV (*Open Source Computer Vision*) adalah *library* yang dimanfaatkan dalam pengolahan citra dinamis secara *real-time*. OpenCV dapat digunakan dalam berbagai bahasa pemograman seperti Python, C++, Java, atau MATLAB. OpenCV memiliki fitur seperti *Feature & Object Detection, Motion Analysis and Object Tracking, Image Filtering, Image Processing*, dan lain-lain. [12]

2.10 Numpy

Numpy adalah *library Python* yang mendukung pengolahan data pada *array* dan matriks multidimensi yang besar. Numpy menyediakan kumpulan fungsi matematika, seperti aljabar linear, transformasi Fourier, pembuatan angka acak, dan lain-lain. *Numpy* bersifat *open source* sehingga banyak dimanfaatkan dalam pengolahan data peneilitian. [13]

2.11 Scikit-learn

Scikit-learn adalah *open source machine learning library* untuk bahasa pemograman Python. Scikit-learn menyediakan fitur seperti *classification*, *regression*, *clustering*, termasuk juga

didalamnya algoritma *support vector machines*, *random forest*, *gradient boosting*, dan lain-lain. [14]

2.12 Pandas

Pandas adalah *library Python* yang mendukung manipulasi data dan analisis. Pandas menyediakan struktur data dan operasi untuk mengolah data numerik maupun time series. Pandas merupakan free software yang dirilis oleh lisensi three-clause BSD yang banyak dimanfaatkan dalam pengolahan data dan penelitian.

2.13 Math

Math merupakan modul yang menyediakan berbagai fungsi matematika yang telah didefinisikan oleh standar Bahasa C. Fungsi ini tidak dapat menggunakan angka yang kompleks

(Halaman ini sengaja dikosongkan)

BAB III
PERANCANGAN SISTEM

Bab ini menjelaskan mengenai perancangan sistem pendeteksi api berdasarkan data video menggunakan optical flow dan klasifikasi SVM.Pada bab ini pula akan dijelaskan gambaran umum sistem dalam bentuk diagram alir.

3.1 Perancangan Data

Data yang digunakan dalam sistem ini diambil dari KMU Fire and Smoke Database, video open source, MIVIA fire dataset, dan video Youtube. Beberapa video terlebih dahulu dipotong agar durasinya tidak melebihi 21 detik kemudian video di-*export* ke dalam format mp4. Contoh frame yang akan diolah dalam sistem adalah

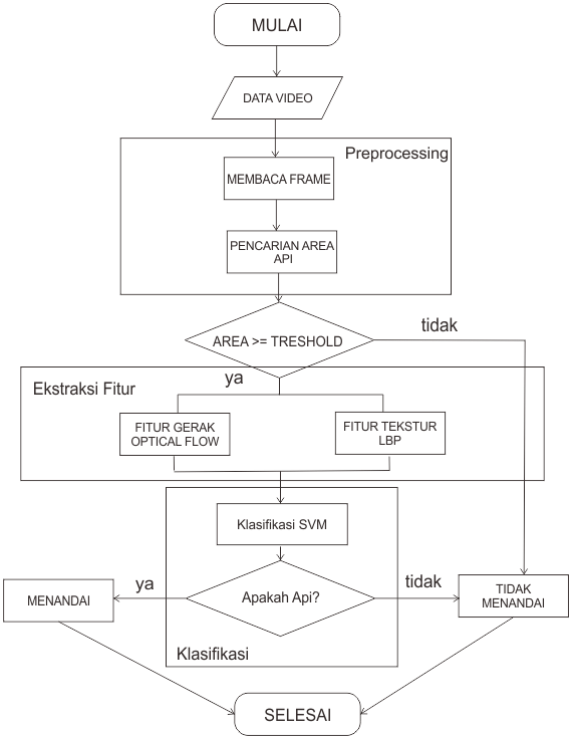
Gambar 3.1 Contoh Frame Api

Gambar 3.2 Contoh Frame Bukan Api

Data dibagi kedalam dua bagian yaitu Dataset untuk melakukan cross validation dan membangun model serta dataset yang tidak pernah ditrain sebelumnya untuk dilakukan pengujian secara langsung

3.2 Desain Umum Sistem

Pada tugas akhir ini akan dilakukan deteksi api berdasarkan sensor visual. Perangkat lunak dibuat dengan menggunakan dataset video yang memiliki frekuensi minimal 20 Hz, kualitas video yang digunakan adalah 240x320 piksel. Sedangkan data pembelajaran adalah berupa data video yang berisi objek api dan objek bukan api.

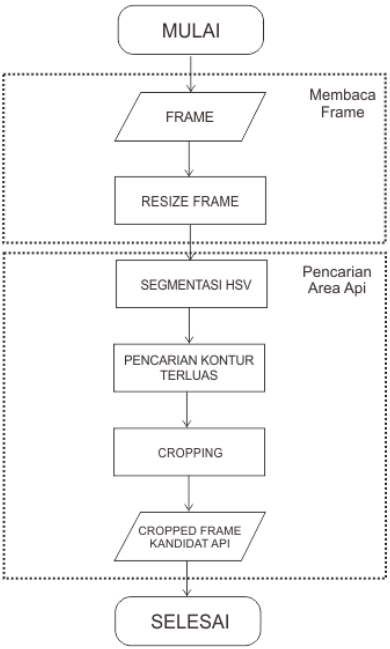


Gambar 3.3 Diagram Alir Sistem yang Dibangun

Rancangan perangkat lunak ini dimulai dengan membaca masukan berupa video. Proses deteksi api dibagi menjadi tiga proses utama yaitu *preprocessing*, ekstraksi fitur dan klasifikasi. Rancangan perangkat lunak secara detail digambarkan pada Gambar 3.3.

3.2.1 Tahap Praproses Data

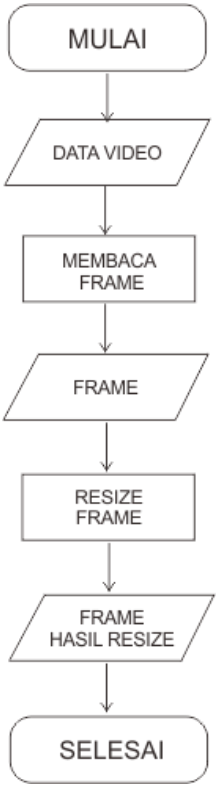
Pada tugas akhir kali ini, praproses data yang dilakukan adalah melakukan perubahan ukuran resolusi pada data video, kemudian segmentasi area api menggunakan *hue, saturation, intensity*. Hasil dari proses segmentasi akan digunakan untuk mendapatkan kontur terbesar dan melakukan *cropping* sebagai area yang akan diobservasi untuk tahap berikutnya. Diagram alir dari proses ditunjukan pada Gambar 3.4.



Gambar 3.4 Diagram Alir Preprocessing

3.2.1.1 Membaca Frame

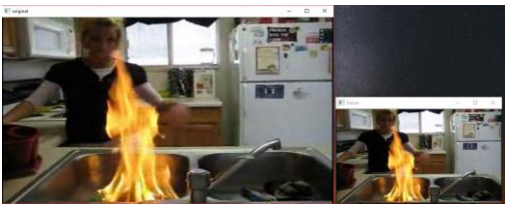
Pada setiap *frame* yang dibaca dilakukan *preprocessing*. *Preprocessing* dimulai dengan mengubah ukuran *frame* menggunakan *gaussian pyramid*, terdiri dari proses *smoothing* dan *downsampling*. Hal ini dilakukan untuk mempercepat proses komputasi. Bagan dari proses ini ditunjukkan oleh



Gambar 3.5 Diagram Alir *Resize Frame*

Masukan dari tahap ini adalah *frame* yang sedang diproses dalam channel R, G, B. Dengan menggunakan metode *gaussian*

pyramid maka didapatkan *frame* yang telah tereduksi kolom dan barisnya menjadi seperempat dari ukuran sebelumnya. Contoh implementasi dari gaussian pyramid ditunjukkan pada Gambar 3.6



(a) (b)

Gambar 3.6 (a) Frame sebelum di-*resize* (b) Frame setelah di-*resize*

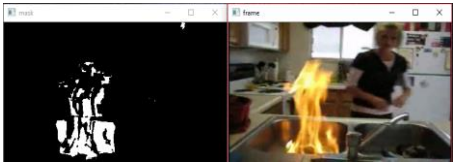
3.2.1.2 Pencarian Area Kandidat Api

Pada tahap deteksi warna digunakan 6 gambar api untuk menentukan nilai warna api. Nilai mean R,G,B dari 6 gambar api adalah *blue* sebesar 12.62, *green* sebesar 78.07 dan *red* sebesar 186.47, kemudian menggunakan Persamaan (2.5) dan (2.6) untuk mendapatkan nilai *hue*, dengan Persamaan (2.7) dan Persamaan (2.8) untuk mendapatkan nilai *saturation* dan *value*, sehingga diperoleh nilai *hue, saturation* dan *value* berturut turut sebesar 11, 239, dan 185.

Proses segmentasi dilakukan dengan cara *masking* mengguakan nilai *hue, saturation, dan value* dengan *threshold hue* sebesar 20, *threshold saturation* 100, dan *threshold value* sebesar 70. *Masking* adalah proses segmentasi biner yang mengubah nilai diluar rentang nilai hue, saturation, value yang ditetapkan menjadi hitam dan yang berada pada rentang tersebut menjadi putih.

Hal yang perlu diperhatikan dalam model warna hsv adalah *hue* memiliki rentang 0-180 yang merupakan representasi sudut sehingga apabila nilai batas minimum *hue* kurang dari 0 atau lebih dari 180 maka perlu dilakukan *masking* ganda. Pada sistem ini nilai batas minimum *hue* mencapai -9 sehingga pada proses *masking* pertama digunakan rentang *hue* 0-31, *saturation* pada rentang 239-255 dan rentang *value* 186-255. Proses selanjutnya, citra biner hasil

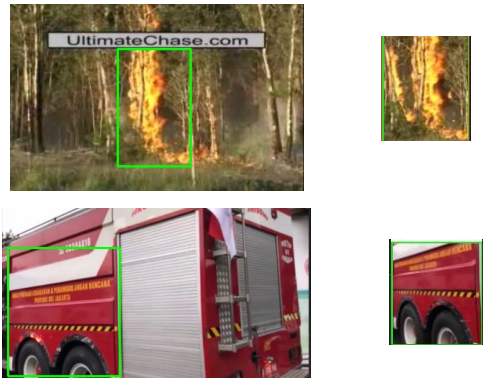
masking pertama digabungkan dengan citra biner hasil *masking* kedua yang diperoleh melalui proses *masking* dengan rentang *hue* 171-180, *saturation* pada rentang 239-255 dan rentang *value* 186-255.Contoh penerapan segmentasi menggunakan masking *hue, saturation, value* terdapat pada Gambar 3.7 berikut



(a) (b)

Gambar 3.7 (a) Frame Api Hasil Segmentasi (b) Frame Api Asli

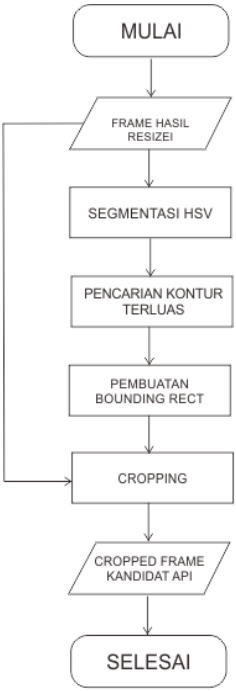
Setelah melakukan segmentasi pada bagian api, maka dilakukan pencarian kontur terluas dari area yang tersegmentasi, kemudian dibuat *bounding rect* berdasarkan kontur terluas tersebut, kemudian dilakukan pemotongan pada frame yang telah di *resize* berdasarkan *bounding rect* tersebut. Proses ini ditunjukkan pada Gambar 3.8



(a) (b)

Gambar 3.8 (a) Pembuatan *bounding rect* berdasarkan kontur terluas (b) Citra hasil *cropping frame* sebagai kandidat api

Diagram alir dari proses pencarian area kandidat api ditunjukkan oleh Gambar 3.9



Gambar 3.9 Diagram Alir Segmentasi HSV

3.2.2 Ekstraksi Fitur

Sebelum melakukan klasifikasi citra yang telah di-*crop*, terlebih dahulu diperlukan berbagai fitur yang diperoleh dari proses ekstraksi fitur. Ekstraksi fitur yang dilakukan meliputi ekstraksi fitur menggunakan *local binary pattern* dan *optical flow*.

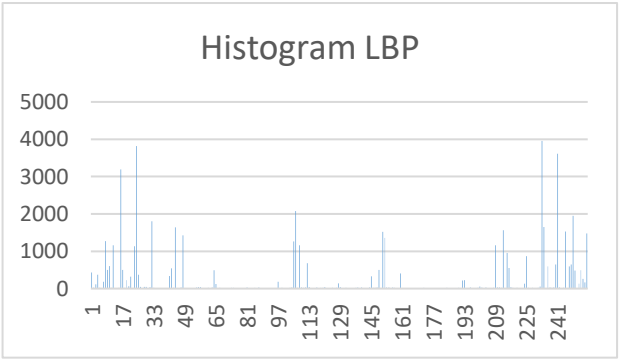
3.2.2.1 Ekstraksi Fitur Tekstur dengan LBP

Pada proses ektraksi fitur, region yang telah ditetapkan sebagai region api, dideteksi teksturnya menggunakan metode

Local Binary Pattern (LBP).Proses perubahan citra dari setiap tahap dijelaskan dalam Gambar 3.10. Sebelum melalui proses LBP citra yang telah dipotong berdasarkan hasil segmentasi terlebih dahulu diubah kedalam citra *grayscale*, kemudian citra *grayscale* akan masuk kedalam tahap LBP.

Metode ini melakukan konvolusi pada setiap 3x3 piksel yang berketetanggaan untuk mencari nilai LBP. Proses berikutnya, nilai keabuan dari citra LBP dijadikan histogram untuk kemudian dijadikan fitur. Sehingga terdapat 256 fitur dari ekstraksi fitur LBP. Berikut adalah citra hasil LBP.

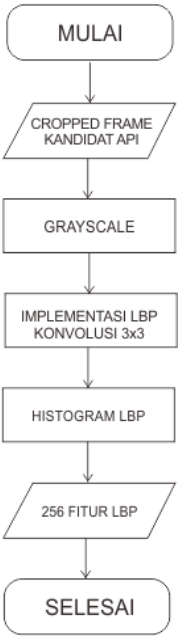
(a) (b) (c)



(d)

Gambar 3.10 Proses Perubahan Citra (a) *cropped frame* (b) citra *grayscale* (c) citra LBP dan (d) histogram LBP

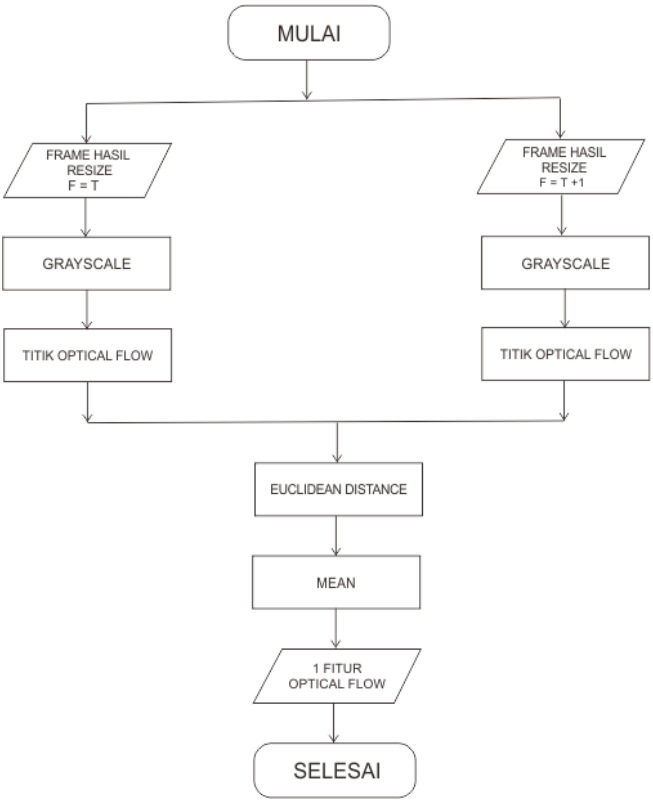
Tahapan dari proses perubahan citra cropping hasil segmentasi hsv hingga menjadi 256 fitur LBP dijelaskan secara bertahap pada Gambar 3.11



Gambar 3.11 Proses Ekstraksi LBP

3.2.2.2 Ekstraksi Fitur Gerak Menggunakan Optical Flow

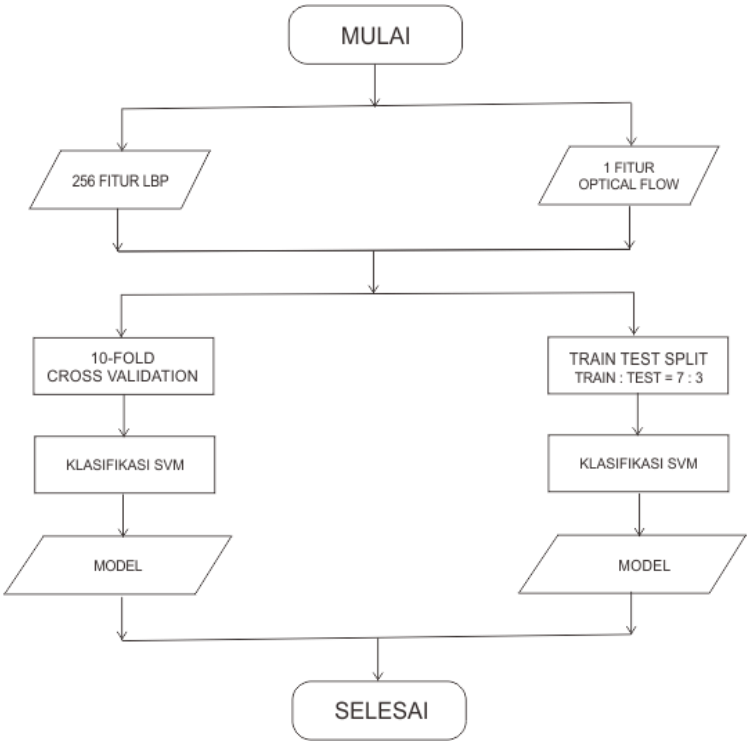
Berdasarkan perhitungan pergerakan menggunakan metode Optical Flow maka akan ditemukan poin untuk diobservasi pergerakannya sehingga kita bisa mengetahui kecepatan gerakan titik tersebut (*vt*). Perhitungan jarak pergerakan suatu titik ke titik yang lain dihitung menggunakan rumus euclidean distance pada Persamaan (2.30). Diagram alir proses ekstraksi fitur optical flow terdapat pada Gambar 3.12



Gambar 3.12 Diagram Alir Optical Flow

3.2.3 Klasifikasi

Hasil Ekstraksi fitur *Optical Flow* dan LBP digunakan dalam tahap klasifikasi. Dari fitur-fitur yang diperoleh dilakukan *training* pada data video untuk video api dan bukan api sehingga dapat mengklasifikasikan data video *testing* kedalam objek api dan bukan api Penjelasan mengenai proses klasifikasi hingga mendapatkan model terdapat pada Gambar 3.13

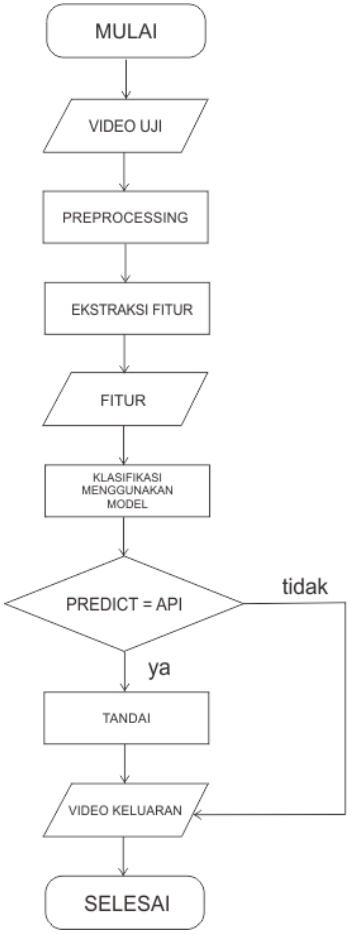


Gambar 3.13 Diagram Alir Klasifikasi

3.2.4 Menandai Region Api

Dalam menandai region api diperlukan model yang dapat menentukan prediksi terhadap data frame video yang baru. Model tersebut didapatkan dengan cara melakukan *cross validation* sebanyak 10fold dan dari setiap iterasi disimpan model dengan akurasi terbaik. Selain itu model didapatkan dengan menggunakan metode pemisahan data latih dan dan data uji dengan perbandingan 3:1. Setelah model tersebut disimpan, model dapat digunakan sebagai acuan dalam menentukan region api pada video baru yang

belum pernah di train sebelumnya. Proses menandai region ai dijelaskan dalam Gambar 3.14 berikut



Gambar 3.14 Diagram Alir Menandai Region Api

Berikut contoh hasil penandaan region api terdapat pada Gambar 3.15

Gambar 3.15 Menandai Region Api

(Halaman ini sengaja dikosongkan)

BAB IV
IMPLEMENTASI

Pada bab ini diuraikan mengenai implementasi perangkat lunak dari rancangan metode yang telah dibahas pada Bab 3 meliputi kode program dalam perangkat lunak. Selain itu, implementasi dari tiap proses, parameter masukan, keluaran, dan beberapa keterangan yang berhubungan dengan program juga dijelaskan.

4.1 Lingkungan Implementasi

Lingkungan implementasi pada tugas akhir ini adalah sebuah *personal computer* (PC). Perangkat PC yang digunakan adalah desktop Lenovo dengan sistem operasi Windows 10 64-bit.

Spesifikasi dari PC yang digunakan memiliki prosesor Intel Core i3-3240 dengan kecepatan 3,4 GHz, *Random Access Memory* (RAM) sebesar 16 GB.

Spesifikasi PC dari sisi perangkat lunak yaitu menggunakan bahasa pemograman Python 3.6, dilengkapi dengan *library* antara lain OpenCV, Numpy, Scikit-learn, Pandas dan Math.

4.3 Implementasi Praproses Data

Pada subbab ini akan dijabarkan implementasi pada tahap pengurangan resolusi citra dan segmentasi dengan menggunakan channel HSI/HSV, proses pencarian kontur terluas, serta proses *cropping* pada segmen area api.

4.3.2 Membaca Frame

Pada sub bab ini dibahas implementasi tahap membaca setiap video sehingga dapat diproses dalam bentuk *frame* dan proses *resize frame*. Masukan dari tahap ini adalah *frame* dan keluarannya adalah *frame* yang telah di*resize* ukurannya. Implementasi dilakukan dengan menggunakan fungsi yang sudah disediakan oleh OpenCV yaitu *pyrDown()* dan ditunjukkan oleh Kode Sumber 4.1

```
1.  if (cap.isOpened()== False):
2.      print("Error opening video stream or file")
3.      while(cap.isOpened()):
4.          ret, frame = cap.read()
5.          if ret == True:
6.              lower_reso = cv2.pyrDown(frame)
7.  cap.release()
8.  cv2.waitKey(1000)
9.  cv2.destroyAllWindows()
```

Kode Sumber 4.1 Reduksi Ukuran Frame

Fungsi *cap.isOpened* berguna untuk mengecek apakah video dapat dibuka. Apabila video dapat dibuka, maka setiap frame akan dibaca sebagai citra dalam variable *frame*. Variabel *lower_reso* akan menyimpan keluaran output yang telah tereduksi

4.3.3 Segmentasi Menggunakan channel HIS/HSV

Frame yang telah di*resize* selanjutanya diproses untuk di segmentasi. Sebelum proses ini dimulai terlebih dahulu dihitung besar range warna api pada *channel red, green* dan *blue*. Pada tugas akhir ini diperoleh nilai *red, green, blue* berturut turut adalah 186, 78 dan 12. Nilai mean tersebut digunakan untuk proses selanjutnya yaitu proses pengubahan *range red, green, blue* kedalam *channel* HSV. Setiap *channel* warna HSV diberikan threshold yang berbeda beda. Dalam penelitian ini digunakan *threshold* 20, 100, 70 masing-masing untuk *channel* warna *hue, saturation,* dan *value*.

Frame yang telah direduksi kemudian diubah kedalam *channel* hsv dan di masking berdasarkan *range hue, saturation, value* dan dengan *threshold* yang telah ditetapkan sebelumnya. Hasil dari proses ini adalah citra biner. Proses tersebut terdapat pada Kode Sumber 4.2.

```
1.  hsv = cv2.cvtColor( np.uint8([[bgr]] ), cv2.COLOR_BGR2HSV)[0][0]
2.
3.  if((hsv[0]-threshh) < 0):
4.      jmask = 2
5.      minHSV1 = np.array([0, hsv[1] - threshs, hsv[2] - threshv])
6.      maxHSV1 = np.array([hsv[0] + threshh, hsv[1] + threshs, hsv[2] + threshv])
7.
8.      minHSV2 = np.array([180 - (threshh - hsv[0]), hsv[1] - threshs, hsv[2] - threshv])
9.      maxHSV2 = np.array([180, hsv[1] + threshs, hsv[2] + threshv])
10.
11. elif((hsv[0]+threshh) > 180):
12.     jmask = 2
13.     minHSV1 = np.array([0, hsv[1] - threshs, hsv[2] - threshv])
14.     maxHSV1 = np.array([180 - (hsv[0]+threshh), hsv[1] + threshs, hsv[2] + threshv])
15.
16.     minHSV2 = np.array([threshh - hsv[0], hsv[1] - threshs, hsv[2] - threshv])
17.     maxHSV2 = np.array([180, hsv[1] + threshs, hsv[2] + threshv])
18. else:
19.     minHSV = np.array([hsv[0] - threshh, hsv[1] - threshs, hsv[2] - threshv])
20.     maxHSV = np.array([hsv[0] + threshh, hsv[1] + threshs, hsv[2] + threshv])
```

Kode Sumber 4.2 Penentuan nilai masking HSV

Kode Sumber 4.2 merupakan proses penentuaan nilai masking hsv. Variable minHSV merupakan nilai batas bawah hsv dan maxHSV adalah batas atas hsv. Pada tugas akhir ini digunakan

masking secara berganda, karena nilai hue dari warna api yaitu merah yang mendekati kuning berada pada pada derajat awal, dan akhir mendekati 0 dan 180, dikarekan HSV memiliki model warna berbentuk kerucut dan melingkar pada channel hue. Masking berganda di implementasikan dalam Kode Sumber 4.3 dibawah ini

```
1.  if (jmask == 2):
2.          masklower = cv2.inRange(hsv, minHSV1, maxHSV1)
3.          maskupper = cv2.inRange(hsv, minHSV2, maxHSV2)
4.          mask = masklower | maskupper
5.      else:
6.          mask = cv2.inRange(hsv, minHSV, maxHSV)
```

Kode Sumber 4.3 Masking HSV

4.3.4 Pencarian Kontur Terluas

Citra biner hasil segmentasi kemudian dihitung luas tiap konturnya untuk mengetahui kontur terluas dari citra yang tersegmentasi. Berdasarkan kontur terluas tersebut kemudian dibuat suatu bounding rect untuk menemukan persegi panjang yang meliputi kontur tersebut. Implementasi pencarian kontur terluas terdapat pada Kode Sumber 4.4

```
  im2,contours,hierarchy = cv2.findContours(mask,
1 cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
2
3       if len(contours) != 0:
4          #find the biggest area
5          c = max(contours, key = cv2.contourArea)
6          x,y,w,h = cv2.boundingRect(c)
```

Kode Sumber 4.4 Implentasi Pencarian Kontur Terluas

Hasil dari proses sebelumnya adalah citra biner yang disimpan dalam variable mask, selanjutnya dalam proses pencarian

kontur digunakan fungsi *findContours()*. Fungsi tersebut menhasilkan keluaran berupa semua kontur dalam citra biner. Pada Kode Sumber 4.4 baris ke-4 menunjukkan c adalah kontur terluas yang diperoleh dengan menggunakan fungsi *max* yang *contourArea* sebagai *key*.

Proses selanjutnya adalah pembuatan persegi panjang pada area kontur terluas. Proses ini di implementasikan menggunakan fungsi *boundingRect*(). Input dari fungsi tersebut adalah citra dengan kontur terluas kemudian menghasilkan *output* berupa nilai koordinat piksel pada ujung kiri atas *boundingRect* serta panjang dan lebar dari *boundingRect*.

4.3.5 Pemotongan Citra

Bounding rect ini kemudian dijadikan acuan dalam memotong frame pada bagian api. Sehingga menghasilkan *cropped frame/ cropped image*. Implementasi pemotongan citra terdapat pada Kode Sumber 4.5

```
1.  if(h >=5 and w >= 5):
2.      crop_img = lower_reso[y:y+h, x:x+w]
```

Kode Sumber 4.5 Pemotongan Citra

Citra dipotong apabila luasan api lebih dari 5x5 pixel. Citra yang dipotong adalah citra dari frame yang telah direduksi sesuai dengan ukuran bounding rect yag diperoleh dari proses segmentasi

4.4 Implementasi Tahap Ekstraksi Fitur

Pada tugas akhir digunakan 2 metode ekstraksi fitur yaitu ekstraksi fitur tekstur menggunakan metode *Local Binary Pattern* dan ekstraksi fitur gerak menggunakan metode *Optical Flow*. Metode LBP menhasilkan 256 fitur yang kemudian digabung dengan 1 fitur dari proses *Optical Flow*.

4.4.1 Ekstraksi Fitur LBP

Data masukan dari proses LBP adalah citra yang telah dipotong pada area api. Kemudian dilakukan proses pengubahan citra menjadi citra keabuan. Citra kemudian diproses untuk diubah menjadi citra LBP. Level keabuan dari citra LBP kemudian dijadikan histogram untuk kemudian dijadikan 256 fitur. Proses ini diimplementasikan dalam Kode Sumber 4.6

```
1. def compute_lbp_histogram(im, radius=1):
2.     im_gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
3.     lbp_img = lbp(im_gray,radius)
4.     hist, _ = np.histogram(lbp_img.ravel(), bins=256)
5.     return np.array(hist).reshape(1,-1)
```

Kode Sumber 4.6 Proses Ekstraksi Fitur LBP

Dalam tugas akhir ini proses mengubah citra grayscale kedalam citra LBP diimplementasikan kedalam Kode Sumber 4.7

```
1.  def lbp(im, r):
2.      new_img = np.zeros(im.shape)
3.      for i in range(im.shape[0]):
4.          for j in range(im.shape[1]):
5.              tmp = ""
6.              for x in range(-r,r+1):
7.                  for y in range(-r,r+1):
8.                      if i+y == i and j+x == j:
9.                          Continue
                        if (j+x >0 and j+x<im.shape[1]) and (i+y>0 and
10. i+y<im.shape[0]):
11.                         if im[i+y,j+x]>=im[i,j]:
12.                             tmp+="1"
13.                         else:
```

```
14.                     tmp+="0"
15.             new_img[i,j] = int(tmp,2)
16.     return new_img
```

Kode Sumber 4.7 Implementasi LBP

Pada Kode Sumber 4.7 digunakan ketenggaan 3x3 sehingga radius yang digunakan adalah 1. Dilakukan konvolusi untuk membandingkan nilai piksel tengah dengan 8 tetangga yang lain, apabila lebih atau sama dengan nilai piksel tengah maka variabel *tmp* akan menyimpan nilai "1", dan apabila kurang dari nilai tengah maka tmp akan menyimpan nilai "0". Setelah kedelapan tetangga sudah dibandingkan maka setiap nilai *tmp* akan dikalikan dengan 2^0 hingga 2^7. Selanjutnya, citra yang telah diubah menjadi citra LBP disimpan dalam variable *new_img.*

4.4.2 Ekstraksi Fitur Optical Flow

Optical Flow bertujuan untuk melacak poin yang bergerak dari suatu frame ke frame berikutnya. Untuk menggunakan *optical flow* terlebih dahulu harus ditemukan poin yang akan diamati pergeraknnya pada frame pertama. Implementasi dari *optical flow* pada frame pertama terdapat pada Kode Sumber 4.8 dibawah ini.

```
1.  feature_params = dict( maxCorners = 100, qualityLevel = 0.3,
    minDistance = 7, blockSize = 7 )
    lk_params = dict( winSize  = (15,15), maxLevel = 2, criteria =
    (cv2.TERM_CRITERIA_EPS |
2.  cv2.TERM_CRITERIA_COUNT, 10, 0.03))
3.
4.  old_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
                    p0 = cv2.goodFeaturesToTrack(old_gray, mask =
5.  None, **feature_params)
6.
7.  dvelodata=pd.DataFrame()
```

Kode Sumber 4.8 Implementasi Optical Flow Pada Frame Pertama

Untuk menentukan poin diatas, digunakan fungsi *cv.goodFeaturesToTrack()*. Poin yang dilacak diperoleh terlebih dahulu dari frame pertama, mendeteksi beberapa titik sudut menggunakan deteksi titik sudut menggunakan fungsi Shi-Tomasi, kemudian secara iteratif melacak titik-titik tersebut menggunakan *optical flow Lucas-Kanade* menggunakan fungsi *cv.calcOpticalFlowPyrLK()*. Variabel *dvelodata* disiapkan untuk menyiapkan jarak dari setiap titik.

```
1   frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
                    p1, st, err =
    cv2.calcOpticalFlowPyrLK(old_gray, frame_gray, p0, None,
2   **lk_params)
3   good_new = p1[st==1]
4   good_old = p0[st==1]
5
6   if(len(good_new) > 0 and len(good_old) > 0):
7       velo=[]
8       for i,(new,old) in enumerate(zip(good_new,good_old)):
9       distance = euclideanDistance(new, old, 2)
10            velo.append(distance)
11      p,q = new.ravel()
12      r,s = old.ravel()
13  dvelo = np.mean(velo)
```

Kode Sumber 4.9 Implementasi Optical Flow Lucas Kanade

Proses optical flow diawali dengan mengubah frame tereduksi menjadi citra keabuan. Dari citra tersebut ditentukan titik yang akan di observasi, titik tersebut didapatkan menggunakan fungsi *goodFeaturesToTrack().* Titik yang telah didapat diseleksi tersebut terlebih dahulu, sehingga didaptkan titik pada area api yang telah disegmentasi. Kemudian ditentukan titk baru dari titik awal yag telah di inisialisasi pada frame pertama menggunakan fungsi *cv.calcOpticalFlowPyrLK()* diambil poin sebelum dan poin

sesudah yang mana memiliki status 1, status satu menandakan bahwa poin selanjutnya dapat dilacak oleh sistem. Apabila poin berikutnya berstatus nol maka poin tersebut tidak dievaluasi. Poin-poin dengan status 1 disimpan dalam variable *good_new* dan *good_old.* Selanjutnya setiap titik pada *good_new* dan *good_old* yang bersesuaian ditentukan jarakanya menggunakan rumus *Euclidean distance* pada Persamaan (2.3) jarak yang didapatkan kemudian di rata rata, karena setiap frame memiliki jumlah poin yang berbeda.

Metode lain yang di implementasikan dalam tugas akhir ini adalah metode Farneback yaitu optical flow dengan mengobservasi semua titik pada area objek api. Implementasi dari metode tersebut terdapat pada.

```
1  def draw_flowme(img,
   flow,colxawal,rowyawal,colxakhir,rowyakhir, step=8):
2
3      h, w = img.shape[:2]
4      y, x = np.mgrid[step/2:h:step, step/2:w:step].reshape(2,-
   1).astype(int)
5      fx, fy = flow[y,x].T
6      lines = np.vstack([x, y, x+fx, y+fy]).T.reshape(-1, 2, 2)
7      lines = np.int32(lines + 0.5)
8      max1= np.int32(max(x))
9      maxx=int(max1/step)
10     kiriatas =int((rowyawal*maxx) + (colxawal+rowyawal))
11     kananatas = int((rowyawal*maxx) + (colxakhir+rowyawal))
12     arr=[]
13     sudah = 0
14     velo=[]
15     while(rowyawal<rowyakhir-1):
16         sudah=1
17         arr = np.append(arr,np.arange(kiriatas,kananatas))
```

```
18          rowyawal = rowyawal+1
19          kiriatas =int((rowyawal*maxx) + (colxawal+rowyawal))
            kananatas = int((rowyawal*maxx) +
20  (colxakhir+rowyawal))
21
22      if(sudah==1):
23          arr = np.int32(arr)
24          data_extractions = (lines[arr]).reshape(-1,2,2)
25          vis = cv2.cvtColor(img, cv2.COLOR_GRAY2BGR)
26          cv2.polylines(vis, data_extractions, 0, (255, 0, 0))
27          distance=0
28
29          for (x1, y1), (x2, y2) in data_extractions:
30              cv2.circle(vis, (x1, y1), 1, (0, 255, 0), -1)
31              distance += pow((y2 - y1), 2)
32              distance += pow((x2 - x1), 2)
33              distance = math.sqrt(distance)
34              velo.append(distance)
35
36      if(sudah==0 and not velo):
37          dvelo = 0.00
38      else:
39          dvelo = np.mean(velo, dtype=np.float64)
40
41      return dvelo
```

Kode Sumber 4.10 Optical Flow Farneback

Pada Kode Sumber 4.10 dihitung terlebih dahulu berapa jumlah grid yang dibutuhkan untuk mengobservasi gerakan suatu frame area api. Posisi grid ditentukan dalam variabel x dan y. Perpindahan titik tersebut disimpan dalam variabel *flow* yang telah

dihitung menggunakan metode farnner back, kemudian dicari titik titik yang merupakan area api yang diperoleh dari proses segmentasi citra. Besar perpindahan dari titik tersebut dijumlahkan kemudian dicari rata-rata nya. Rata rata perpindahan tersebut dijadikan sebagai satu fitur *optical flow*.

4.5 Implementasi Tahap Klasifikasi

Pada tugas akhir ini klasifikasi dilakukan dengan metode Support Vector Machine. Terdapat 2 metode yang dilakukan dalam mendapatkan model, yaitu pembagian data kedalam 10 cross validation dan membagi data dalam data latih dan data uji dengan perbandingan 7:3. Implementasi dari tahap klasifikasi dituliskan dalam Kode Sumber 4.11

```
1. kelas = re.findall("ID_(\d+).mp4", filevideo)
2.  histogram = compute_lbp_histogram(crop_img)
3. histogram = np.concatenate((histogram, dvelo), axis=None)
4.  histogram = histogram.reshape(1,-1)
5. histogram = pd.DataFrame(histogram)
6. classdata = classdata.append(kelas, ignore_index=True)
7.  dataset = dataset.append(histogram, ignore_index=True)
```

Kode Sumber 4.11 Implementasi Tahap Klasifikasi

Pada setiap video diambil kelas dari label pada video tersebut. Dataset merupakan variable yang menyimpan 256 fitur LBP dan 1 fitur optical flow. Selanjutnya data akan memasuki tahap klasifikasi SVM

Metode pertama adalah penggunaan cross validation untuk mendapatkan model. Model yang disimpan adalah model dengan akurasi terbaik. Selain itu, model diperoleh dari pembagian dataset menjadi data latih dan data uji dengan perbandingan 7:3. Klasifikasi SVM dengan pembagian data latih dan data uji dengan perbandingan 7:3 diimplementasikan dalam Kode Sumber 4.12

```
1. def SVM_nocrossval(dataset,classdata):
2.     X_train =[]
```

```
3.    X_test = []
4.    y_train = []
5.    y_test = []
6.    X_train, X_test, y_train, y_test = train_test_split(dataset, classdata, test_size=0.3)
7.    clf = svm.SVC (C=1.0, kernel='linear',gamma=0.001, tol=0.001, max_iter=-1)
8.    model = clf.fit(X_train, y_train)
9.    model_result = model.predict(X_test)
10.   filename = 'model_nocross_poly.sav'
11.   pickle.dump(model, open(filename, 'wb'))
12.   old_ac = (float(sm.accuracy_score(model_result, y_test)) * 100)
```

Kode Sumber 4.12 SVM Tanpa Cross Validation

Setelah melakukan proses training maka model disimpan dalam file berbentuk .sav. Implementasi klasifikasi SVM tanpa menggunakan croos validation terdapat pada Kode Sumber 4.13

```
1.  def SVM_model(dataset, classdata):
2.      X_train =[]
3.      X_test = []
4.      y_train = []
5.      y_test = []
6.      list_akurasi = []
7.      new_ac =0
8.
9.      skf = StratifiedKFold(n_splits=10)
10.     clf = svm.SVC (C=1.0, kernel='poly',gamma=0.001, tol=0.001, max_iter=-1)
11.     for train_index, test_index in skf.split(dataset, classdata):
12.         X_train = [dataset.loc[i,:] for i in train_index]
13.         X_test = [dataset.loc[i,:] for i in test_index]
```

```
        y_train, y_test = classdata.loc[train_index,:],
14. classdata.loc[test_index,:]
15.     y_train = pd.DataFrame(y_train)
16.     y_test = pd.DataFrame(y_test)
17.     model = clf.fit(X_train, y_train)
18.     model_result = model.predict(X_test)
        old_ac = (float(sm.accuracy_score(model_result, y_test)) *
19. 100)
20.     list_akurasi.append(old_ac)
21.     if(old_ac> new_ac):
22.         new_ac = old_ac
23.         # save the model to disk
24.         filename = 'model_ubahcross_poly.sav'
25.         pickle.dump(model, open(filename, 'wb'))
26.     print(old_ac)
27.
28.   print(np.mean(list_akurasi))
```

Kode Sumber 4.13 Implementasi SVM dengan Cross Validation

4.6 Implementasi Tahap Menandai region Api

Dengan menggunakan model yang sudah dismpan sebelumnya maka, setelah melakukan proses preprocessing dan ekstraksi fitur pada video uji yang baru, maka sistem dapat langsung memprediksi ada tidaknya api dalam video baru tersebut. Apabila suatu video diprediksi sebagai apai, maka *bounding rect* yang telah dibuat ditampilkan. Apabila frame tersebut tidak mengandung api, maka tidak akan ada *bounding rect* dalam frame. Implementasi tahap menandai region api terdapat pada Kode Sumber 4.14

```
1. filename = 'model_ubahcross_poly.sav'
2. loaded_model = pickle.load(open(filename, 'rb'))
3. hasil = loaded_model.predict(histogram)
```

```
4.  hasilnya = str(hasil[0])
5.  if(hasilnya == "1" ):
6.          cv2.rectangle(lower_reso,(x,y),(x+w,y+h),(0,255,0),2)
7.  cv2.imshow('frame',lower_reso)
```

Kode Sumber 4.14 Implementasi menandai Region Api

BAB V
UJI COBA DAN EVALUASI

Dalam bab ini dibahas mengenai hasil uji coba sistem yang telah dirancang dan dibuat. Uji coba dilakukan untuk mengetahui kinerja sistem dengan lingkungan uji coba yang telah ditentukan.

5.1 Lingkungan Uji Coba

Lingkungan implementasi pada tugas akhir ini adalah sebuah *personal computer* (PC). Perangkat PC yang digunakan adalah desktop Lenovo dengan sistem operasi Windows 10 64-bit.

Spesifikasi dari PC yang digunakan memiliki prosesor Intel Core i3-3240 dengan kecepatan 3,4 GHz, *Random Access Memory* (RAM) sebesar 16 GB.

Spesifikasi PC dari sisi perangkat lunak yaitu menggunakan bahasa pemograman Python 3.6, dilengkapi dengan *library* antara lain OpenCV, Numpy, Scikit-learn, Pandas dan Math.

5.2 Dataset

Data yang digunakan untuk uji coba implementasi deteksi api berbasis data video menggunakan metode *optical flow* dan *support vector machines* adalah potongan video dari berbagai sumber berbeda. Kualitas video yang digunakan adalah video dengan ukuran 854x480 piksel dan memiliki channel R,G,B dan memiliki durasi tidak lebih dari 21 detik. Video berisi berbagai kejadian. Terdapat dua jenis video, yaitu video objek api dan video tanpa objek api. Pada video bukan objek api, terdapat beberapa kejadian yang mirip dengan adanya objek api seperti sorot lampu mobil pada malam hari, video jalanan kota dengan beberapa mobil merah dan objek objek berwarna merah serta beberapa video mobil pemadam kebakaran.

Jumlah video yang digunakan dalam tugas akhir ini adalah 56 video yaitu 31 video untuk membuat model yang terdiri dari 16 video api dan 15 video bukan api, dan 25 video yang terdiri dari 8909 frame yaitu 5217 frame api dan 3692 frame bukan api untuk

video uji yang tidak pernah memasuki proses training sebelumnya. Contoh video kejadian dapat dilihat Gambar 5.1

Gambar 5.1 Contoh Video Dataset

5.3 Alur Uji Coba

Pada subbab ini akan dijelaskan alur kerja dari sitem deteksi api dari tahap preprocessing, ekstraksi fitur, hingga klasifikasi

5.3.1 Preprocessing

Pada tahap ini, frame yang telah dibaca direduksi terlebih dahulu kemudian dilakukan segmentasi warna HSI, kemudian dicari kontur dengan area terluas, kemudian dilakukan *cropping* pada area api untuk menjadi inputan pada ekstrasi fitur LBP. *Frame* masukan pada tahap ini ditunjukkan pada tahapan *preprocessing*

dijelaskan pada Gambar 5.2 dan Gambar 5.3 adalah tahap *resize* ukuran *frame*. Pada Gambar 5.4 adalah proses segmentasi HIS/HSV. Citra hasil *cropping* kandidat api terdapat pada Gambar 5.5

Gambar 5.2 *Frame* Masukan Tahap *Preprocessing*

Gambar 5.3 *Resize* Ukuran *Frame*

Gambar 5.4 Hasil Tahap Segemnetasi HIS/HSV

Gambar 5.5 Citra Hasil *Cropping* Kandidat Api

5.3.2 Ekstraksi Fitur

Pada tahap ini, frame yang telah di *crop* pada bagian api akan diubah kedalam citra LBP kemudian tingkat keabuaanya dijadikan histogram untuk kemudian dijadikan fitur. Ekstraksi fitur lainnya adalah optical flow, dimana optical flow akan memilih poin untuk dijadikan acuan dalam mencari kecepatan gerakan dari suatu objek yang diamati. Citra hasil ekstraksi fitur LBP ditunjukkan dalam Gambar 5.6, sementara citra ilustrasi ekstraksi fitur optical flow ditunjukkan oleh Gambar 5.7

Gambar 5.6 Citra Hasil LBP

(a) (b)

Gambar 5.7 Ekstraksi Fitur *Optical Flow* (a) *Lucas Kanade* (b) *Farneback*

5.3.3 Klasifikasi SVM

Pada proses ini fitur fitur yang telah didapatkan diklasifikasikan berdasarkan model yang telah dibuat pada tahap *training* sebelumnya sehingga sistem dapat menandai region api. Region api ditandai berdasarkan *bounding rect* yang diperoleh pada tahap sebelumnya. Apabila sistem mengklasifikasikan fitur sebagai kelas api maka *bounding rect* akan ditampilkan apabila tidak, maka akan dikeluarkan sebagai video keluaran. Proses ini ditunjukkan pada Gambar 5.8

Gambar 5.8 Hasil Klasifikasi SVM

5.4 Skenario Uji Coba

Pada subbab ini dijelaskan mengenai skenario uji coba yang telah dilakukan. Telah dilakukan beberapa skenario uji coba, diantaranya, yaitu:

1. Perbandingan hasil akurasi pada pembuatan model menggunakan *cross validation* dan menggunakan pembagian data latih dan data uji dengan perbandingan 7:3. Perbandingan akurasi pada kernel linear, rbf, sigmoid dan polynomial 2 pada 10 fold, menggunakan fitur LBP dan *optical flow Lucas Kanade*
2. Perbandingan hasil akurasi pada pembuatan model menggunakan *cross validation* dan menggunakan pembagian data latih dan data uji dengan perbandingan 7:3. Perbandingan akurasi pada kernel linear, rbf, sigmoid

dan polynomial 2 pada 10 fold, menggunakan fitur LBP dan *optical flow Farneback*

3. Perbandingan hasil akurasi pada pembuatan model menggunakan *cross validation* dan menggunakan pembagian data latih dan data uji dengan perbandingan 7:3. Perbandingan akurasi pada kernel linear, rbf, sigmoid dan polynomial 2 pada 10 fold, menggunakan fitur LBP
4. Perbandingan nilai akurasi, presisi, dan *recall* pada 25 video yang belum pernah ditrain pada kernel linear dan polynomial 2 dengan fitur LBP dan *Optical Flow Lucas Kanade.*
5. Perbandingan nilai akurasi, presisi, dan *recall* pada 25 video yang belum pernah ditrain pada kernel linear dan polynomial 2 dengan fitur LBP dan *Optical Flow Farne Back.*
6. Perbandingan nilai akurasi, presisi, dan *recall* pada 25 video yang belum pernah ditrain pada kernel linear dan polynomial 2 dengan fitur LBP

5.4.1 Skenario Uji Coba 1

Pada subbab ini dijelaskan cara pembuatan model pada sistem pendeteksi api. Model dibuat dengan dua metode yaitu *cross validation* dengan menggunakan 10 *fold* dan metode pembagian data latih dan data uji dengan perbandingan 7:3. Hasil dari sistem ini adalah akurasi pada tiap proses pembuatan model. Disini digunakan empat kernel SVM sebagai pembanding yaitu kernel linear, sigmoid, rbf dan polynomial 2. Fitur yang digunakan dalam skenario uji coba ini adalah fitur LBP dan *Optical Flow Lucas Kanade*. Akurasi pada setiap *fold* dan akurasi rata rata *fold* ditunjukkan pada dan Tabel 5.1

Tabel 5.1 Hasil 10 *Cross Validation* Fitur LBP + OP *Lucas Kanade*

	Akurasi (%)			
Fold ke-	**Polinomial 2**	**Linear**	**Sigmoid**	**RBF**
1	99.12	97.56	58.21	58.42
2	99.40	94.31	58.33	58.54
3	75.61	87.6	58.33	59.96
4	95.53	96.75	63.21	63.22
5	99.39	99.6	58.25	58.25
6	100.00	100	58.25	58.25
7	89.61	98.17	58.25	58.25
8	87.37	78.82	58.25	57.43
9	95.93	94.3	58.25	58.25
10	98.57	95.93	58.25	58.25
Rata – rata	**94.06**	**94.30**	**58.76**	**58.88**

Uji coba pembuatan model menggunakan kernel SVM polinomial 2, linear, sigmoid dan rbf dengan metode pembagian data latih dan dan data uji dengan perbandingan 7:3 dengan fitur campuran LBP dan *optical flow Lucas Kanade* adalah seperti pada Tabel 5.2

Tabel 5.2 Akurasi *Train Test Split* Fitur LBP dan OP *Lucas Kanade*

Kernel	**Akurasi (%)**
Polinomial 2	99.12
Linear	98.85
Sigmoid	57.15
RBF	64.81

5.4.2 Skenario Uji Coba 2

Pada subbab ini dijelaskan cara pembuatan model pada sistem pendeteksi api. Model dibuat dengan dua metode yaitu

croos validation dengan menggunakan 10 *fold* dan metode pembagian data training dan data testing dengan perbandingan 7:3. Hasil dari sistem ini adalah akurasi pada tiap proses pembuatan model. Disini digunakan empat kernel SVM sebagai pembanding yaitu kernel linear, sigmoid, rbf dan polinomial 2. Fitur yang diguakan dalam skenario uji coba ini adalah LBP dan *Optical Flow Farneback*. Akurasi pada setiap *fold* dan akurasi rata rata *fold* ditunjukkan pada Tabel 5.3

Tabel 5.3 Hasil 10 Cross Validation Fitur LBP + OP Farneback

	Akurasi (%)			
Fold ke-	**Polinomial 2**	**Linear**	**Sigmoid**	**RBF**
1	48.88	53.67	49.12	56.29
2	81.31	80.23	41.78	56.29
3	59.00	87.18	57.45	56.37
4	96.99	97.14	34.21	54.59
5	90.03	93.97	54.79	56.26
6	95.21	94.51	45.98	55.87
7	91.88	83.46	38.33	63.06
8	61.95	77.03	41.69	67.59
9	77.8	84.61	44.7	57.77
10	86.62	84.53	42.23	57.3
Rata – rata	**78.97**	**83.63**	**45.03**	**58.14**

Uji coba pembuatan model menggunakan kernel SVM polynomial 2, linear, sigmoid dan rbf menggunakan campuran LBP dan *optical flow Farneback* dengan metode pembagian data latih dan dan data uji dengan perbandingan 7:3 adalah seperti pada Tabel 5.4

Tabel 5.4 Akurasi *Train Test Split* Fitur LBP dan OP *Farneback*

Kernel	**Akurasi (%)**
Polinomial 2	98.12
Linear	95.52
Sigmoid	47.69
RBF	68.91

5.4.3 Skenario Uji Coba 3

Pada subbab ini dijelaskan cara pembuatan model pada sistem pendeteksi api. Model dibuat dengan dua metode *yaitu croos validation* dengan menggunakan 10-*fold* dan metode pembagian data latih dan data uji dengan perbandingan 7:3.

Tabel 5.5 Hasil 10 *Cross Validation* Fitur LBP

	Akurasi (%)			
Fold ke-	**Polinomial 2**	**Linear**	**Sigmoid**	**RBF**
1	43.65	44.68	58.21	58.42
2	84.57	84.47	58.33	58.54
3	80.43	92.00	58.33	59.96
4	97.84	86.55	63.21	63.21
5	98.59	98.21	58.24	58.25
6	99.06	99.34	58.24	58.25
7	86.16	79.75	58.24	58.25
8	66.1	89.64	58.24	57.43
9	81.17	84.84	58.24	58.25
10	84.82	92.27	58.24	58.25
Rata – rata	**82.24**	**85.18**	**58.75**	**58.88**

Hasil dari sistem ini adalah akurasi pada tiap proses pembuatan model. Disini digunakan empat kernel SVM sebagai

pembanding yaitu kernel linear dan polynomial. Fitur yang diguakan dalam skenario uji coba ini adalah LBP. Akurasi pada setiap fold dan akurasi rata rata fold ditunjukkan pada Tabel 5.5

Uji coba pembuatan model menggunakan kernel SVM polynomial 2, linear, sigmoid dan rbf menggunakan fitur LBP dengan metode pembagian data latih dan dan data uji dengan perbandingan 7:3 adalah seperti pada

Tabel 5.6 Akurasi *Train Test Split* Fitur LBP

Kernel	**Akurasi (%)**
Polinomial 2	97.96
Linear	96.05
Sigmoid	57.56
RBF	62.64

5.4.4 Skenario Uji Coba 4

Skenario uji coba 4 adalah perbandingan nilai akurasi, presisi, dan recall pada 25 video yang belum pernah memasuki tahap *training*. Model yang digunakan adalah model dengan kernel SVM linear dan polinomial 2 dengan fitur LBP dan *Optical Flow Lucas Kanade*. *Confusion matrix* dari ujicoba ini telah dijelaskan dalam Tabel 2.1. Terdapat 4 model yang di uji cobakan pada video baru, yaitu untuk setiap kernel, terdapat 2 jenis model yaitu model dari hasil *cross validation* dan *train test split.*

Hasil uji coba dari model yang dibuat dengan metode *cross validation* menggunakan kernel SVM polinomial 2, dengan fitur gabungan antara *Local Binary Pattern* dan *Optical Flow Lucas Kanade* ditunjukkan pada Tabel 5.7

Tabel 5.7 *Confusion Matrix* Fitur LBP + OP *Lucas Kanade* Metode *Cross Validation* Kernel Polinomial 2

		Kelas Prediksi	
		Api	Bukan Api
Kelas sebenarnya	Api	4787	430
	Bukan Api	303	3389

Hasil uji coba pengujian video menggunakan model yang dibuat dengan metode *cross validation* menggunakan kernel SVM linear, dengan fitur gabungan antara *Local Binary Pattern dan Optical Flow Lucas Kanade* ditunjukkan pada Tabel 5.8

Tabel 5.8 *Confusion Matrix* Fitur LBP + OP *Lucas Kanade* Metode *Cross Validation* Kernel Linear

		Kelas Prediksi	
		Api	Bukan Api
Kelas sebenarnya	Api	5197	20
	Bukan Api	318	3374

Hasil uji coba pengujian video menggunakan model yang dibuat dengan metode pembagian data latih dan data uji dengan perbandingan 7:3, menggunakan kernel SVM polinomial 2, dengan fitur gabungan antara *Local Binary Pattern* dan *Optical Flow Lucas Kanade* ditunjukkan pada Tabel 5.9

Tabel 5.9 *Confusion Matrix* Fitur LBP + OP *Lucas Kanade* Metode *Cross Validation* Kernel Polinomial 2

		Kelas Prediksi	
		Api	Bukan Api
Kelas sebenarnya	Api	4866	351
	Bukan Api	300	3392

Hasil uji coba pengujian video menggunakan model yang dibuat dengan metode pembagian data latih dan data uji dengan perbandingan 7:3, menggunakan kernel SVM linear, dengan fitur gabungan antara *Local Binary Pattern* dan *Optical Flow Lucas Kanade* ditunjukkan pada Tabel 5.10

Tabel 5.10 *Confusion Matrix* Fitur LBP + OP *Lucas Kanade* Metode *Cross Validation* Kernel Linear

		Kelas Prediksi	
		Api	Bukan Api
Kelas sebenarnya	Api	5203	14
	Bukan Api	346	3346

Besar nilai *recall*, *precision*, dan akurasi dari ujicoba 4 berdasarkan *confusion matrix* sebelumnya ditunjukkan pada Tabel 5.11

Tabel 5.11 Hasil Uji Coba Skenario 4

Model	Recall (%)	Precision (%)	Akurasi (%)
Cross Validation (polinomial 2)	91.76	94.05	91.77
Cross Validation (linear)	99.62	94.23	96.21
Train Test Split (polinomial 2)	93.27	94.19	92.69
Train Test Split (linear)	99.73	93.76	95.96

5.4.5 Skenario Uji Coba 5

Skenario uji coba 5 adalah perbandingan akurasi, precision, dan recall pada 25 video yang belum pernah ditrain pada kernel SVM linear dan polinomial 2 dengan fitur LBP dan *Optical Flow Farneback*. *Confusion matrix* dari ujicoba ini telah diterangkan dalam Tabel 2.1.

Terdapat 4 model yang di uji cobakan pada video baru, yaitu untuk setiap kernel, terdapat 2 jenis model yaitu model dari hasil *cross validation* dan *train test split.*

Hasil uji coba dari model yang dibuat dengan metode *cross validation* menggunakan kernel SVM polinomial 2, dengan fitur gabungan antara *Local Binary Pattern* dan *Optical Flow Farneback* ditunjukkan pada Tabel 5.12

Tabel 5.12 *Confusion Matrix* Fitur LBP + OP *Farneback* Metode *Cross Validation* Kernel Polinomial 2

		Kelas Prediksi	
		Api	Bukan Api
Kelas sebenarnya	Api	5008	209
	Bukan Api	1311	2381

Hasil uji coba pengujian video menggunakan model yang dibuat dengan metode cross validation menggunakan kernel SVM linear, dengan fitur gabungan antara *Local Binary Pattern* dan *Optical Flow Farneback* ditunjukkan pada Tabel 5.13

Tabel 5.13 *Confusion Matrix* Fitur LBP + OP *Farneback* Metode *Cross Validation* Kernel Linear

		Kelas Prediksi	
		Api	Bukan Api
Kelas sebenarnya	Api	5080	137
	Bukan Api	1331	2361

Hasil uji coba pengujian video menggunakan model yang dibuat dengan metode pembagian data latih dan data uji dengan perbandingan 7:3, menggunakan kernel SVM polinomial 2, dengan fitur gabungan antara *Local Binary Pattern* dan *Optical Flow Farneback* ditunjukkan pada Tabel 5.14

Tabel 5.14 *Confusion Matrix* Fitur LBP + OP *Farneback* Metode *Train Test Split* Kernel Polinomial

		Kelas Prediksi	
		Api	Bukan Api
Kelas sebenarnya	Api	5061	156
	Bukan Api	1329	2363

Hasil uji coba dari model yang dibuat dengan metode pembagian data latih dan data uji dengan perbandingan 7:3, menggunakan kernel SVM linear, dengan fitur gabungan antara *Local Binary Pattern* dan *Optical Flow Farneback* ditunjukkan pada

Tabel 5.15 *Confusion Matrix* Fitur LBP + OP *Farneback Metode Train Test Split* Kernel Linear

		Kelas Prediksi	
		Api	Bukan Api
Kelas sebenarnya	Api	4986	231
	Bukan Api	1316	2376

Besar nilai *recall*, *precision*, dan akurasi dari ujicoba 5 berdasarkan *confusion matrix* sebelumnya ditunjukkan pada Tabel 5.11

Tabel 5.16 Hasil Uji Coba Skenario 5

Model	Recall (%)	Precision (%)	Akurasi (%)
Cross Validation (polinomial 2)	95.99	79.25	82.94
Cross Validation (linear)	97.37	79.24	83.52
Train Test Split (polinomial 2)	97.01	79.20	83.33
Train Test Split (linear)	95.57	79.12	82.64

5.4.6 Skenario Uji Coba 6

Skenario uji coba 6 adalah perbandingan akurasi, precision, dan recall pada 25 video yang belum pernah ditrain pada kernel SVM linear dan polinomial 2 dengan fitur LBP. *Confusion matrix* dari ujicoba ini telah diterangkan dalam Tabel 2.1. Terdapat 4 model yang di ujicobakan pada video baru. Hasil uji coba dari model yang dibuat dengan metode *cross validation* menggunakan

kernel SVM polinomial 2, dengan fitur *Local Binary Pattern* ditunjukkan pada Tabel 5.17

Tabel 5.17 *Confusion Matrix* Fitur LBP *Cross Validation* Kernel Polinomial 2

		Kelas Prediksi	
		Api	Bukan Api
Kelas sebenarnya	Api	4459	758
	Bukan Api	442	3250

Hasil uji coba dari model yang dibuat dengan metode cross validation menggunakan kernel SVM linear, dengan fitur *Local Binary Pattern* ditunjukkan pada Tabel 5.18

Tabel 5.18 *Confusion Matrix* Fitur LBP Metode *Cross Validation* Kernel Linear

		Kelas Prediksi	
		Api	Bukan Api
Kelas sebenarnya	Api	4829	388
	Bukan Api	572	3120

Hasil uji coba dari model yang dibuat dengan metode pembagian data latih dan data uji dengan perbandingan 7:3, menggunakan kernel SVM polinomial 2, dengan fitur *Local Binary Pattern* ditunjukkan pada Tabel 5.19

Tabel 5.19 *Confusion Matrix* Fitur LBP Metode *Train Test Split* Kernel Polinomial 2

		Kelas Prediksi	
		Api	Bukan Api
Kelas sebenarnya	Api	4487	730
	Bukan Api	477	3215

Hasil uji coba dari model yang dibuat dengan metode pembagian data latih dan data uji dengan perbandingan 7:3, menggunakan kernel SVM linear, dengan fitur *Local Binary Pattern* ditunjukkan pada Tabel 5.20

Tabel 5.20 *Confusion Matrix* Fitur LBP Metode *Train Test Split* Kernel Linear

		Kelas Prediksi	
		Api	Bukan Api
Kelas sebenarnya	Api	4834	383
	Bukan Api	574	3118

Besar nilai *recall*, *precision*, dan akurasi dari ujicoba 6 berdasarkan *confusion matrix* sebelumnya ditunjukkan pada Tabel 5.21

Tabel 5.21 Hasil Uji Coba Skneario 6

Model	Recall (%)	Precision (%)	Akurasi (%)
Cross Validation (polinomial 2)	85.47	90.98	86.53
Cross Validation (linear)	92.56	89.41	89.22
Train Test Split (polinomial 2)	93.27	94.19	92.69
Train Test Split (linear)	92.66	89.39	89.26

5.5 Analisis Hasil Uji Coba

Pada skenario uji coba 4-6 terdapat beberapa video yang kurang terdeteksi dengan baik, baik itu pada kernel linear maupun kernel polinomial 2. Video tersebut antara lain adalah video kemacetan jalan raya yang direkam dari sisi atas, dan video keadaan jalan raya pada suasana malam hari. Berdasarkan hasil uji coba, video pada Gambar 5.9 (a) hanya memperoleh nilai true

negative sebesar 46-56 % saat diklasfikasi berdasarkan fitur LBP dan optical flow Lucas Kanade, semsntara apabila diklasifikasi menggunakan fitur LBP dan optical Farneback ataupun hanya menggunakan fitur LBP video tersebut memperoleh nilai true negatve sebesar 22-23% . Video kedua, yaitu video yang terdapat pada Gambar 5.9 (b) tidak mampu dideteksi dengan baik apabila menggunakan fitur LBP saja. Video tersebut memperoleh nilai true negative sebesar 22-23%. Dengan penambahan fitur optical flow Farneback pada video tersebut nilai true negative juga tidak mengalami perubahan signifikan yaitu berada pada rentang nilai sebesar 23-24%. Kedua video tersebut memiliki karakteristik berada pada rentang warna hsv api yaitu warna kuning hingga merah, serta memilki tekstur mirip api untuk warna lampu kendaraan pada malam hari dan memiliki pola gerak yang minim seperti api.

(a) (b)

Gambar 5.9 Hasil Misklasifikasi (a) Video Kejadian Jalan Macet (b) Video Kejadian Jalan Suasana Malam Hari

Pada uji coba skenario 4-6, hanya uji coba skenario 6 yang kurang mampu mendeteksi video api dengan baik, yaitu uji coba deteksi api menggunakan fitur LBP, sehingga penentuan adanya api hanya didasarkan pada tekstur pada area yang ditentukan sebagai area kandidat api (*cropped frame*). Video pertama adalah video kejadian kebakaran pada kereta api, video kedua adalah video api yang direkam dalam jarak dekat. Video pertama seperti ditunjukkan pada Gambar 5.10 (a) hanya memperoleh nilai true

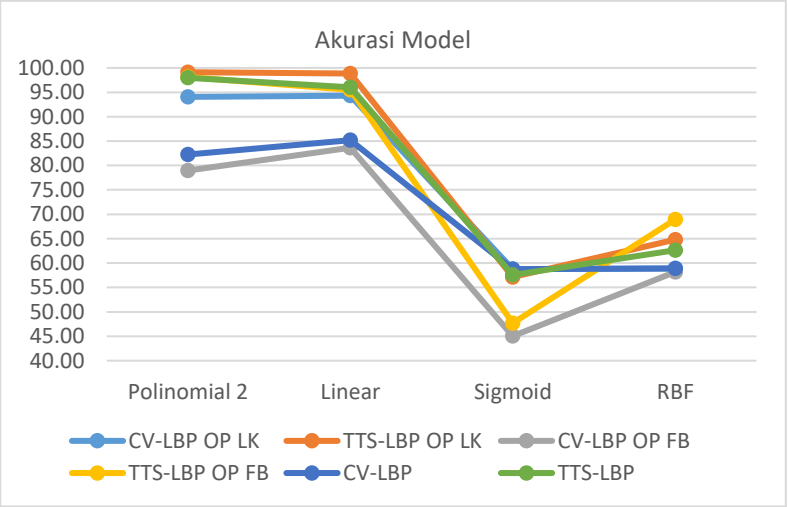
positive sebesar 9,35% sementara video kedua yang ditunjukkan pada Gambar 5.10 (b) memperoleh nilai true positive sebesar 35%. Penggunaan fitur LBP saja pada video ini memungkinkan terjadinya kesalahan karena saat area dipotong dalam bentuk persegi panjang terdapat potongan bagian lain yang bukan area api, seperti gerbong kereta pada video pertama dan batu besar ditengah api pada video kedua. Dua hal tersebut memilki tekstur yang berbeda dengan api, namun cukup dominan dalam *crooped frame* yang dijadikan ekstraksi fitur LBP. Hal inilah yang memungkinkan terjadinya kesalahan deteksi saat hanya menggunakan fitur LBP.

(a) (b)

Gambar 5.10 Hasil Misklasifikasi (a) Video Kejadian Kebakaran Kereta Api (b) Video Api

Dari skenario hasil uji coba yang telah dilakukan, beberapa parameter, metode pembuatan model dan metode optical flow memberikan pengaruh terhadap hasil deteksi. Parameter yang digunakan antara lain kernel svm, yaitu kernel polynomial 2, rbf, sigmoid dan kernel linear, untuk metode pembuatan model diujicobakan dua jenis metode yaitu *cross validation* dengan 10 fold dan pembagian data latih dan data uji dengan perbandingan 7:3, sementara untuk metode optical flow, di ujicobakan 2 jenis metode yaitu sparse optical flow menggunakan metode Lucas Kanade dan dense optical flow menggunakan metode Farneback. Ujicoba dilakukan dengan menggunakan akurasi untuk uji coba model dan menggunakan nilai akurasi, presisi, dan *recall*.

Untuk ujicoba pembuatan model yaitu ujicoba 1-4 menunjukkan bahwa akurasi pembuatan model mendapatkan nilai yang tinggi pada kernel linear dan polynomial, berbeda jauh dengan akurasi model pada kernel sigmoid dan RBF. Hal ini ditunjukkan pada Gambar 5.11

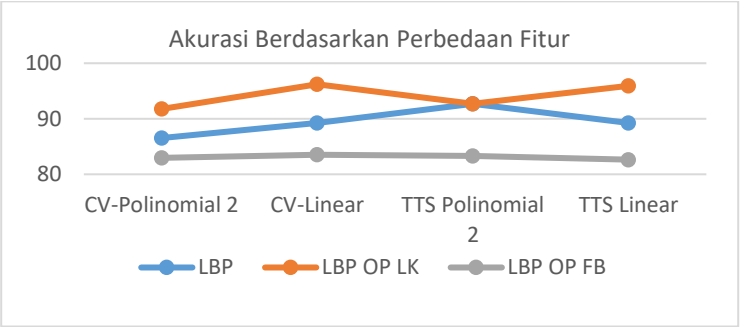


Gambar 5.11 Grafik Akurasi Model

Hasil ujicoba 2 dan 5 yang ditunjukkan pada Gambar 5.12 bahwa perbedaan metode optical flow menghasilkan akurasi yang berbeda. Optical flow menggunakan sparse optical flow memiliki akurasi yang lebih tinggi jika dibandingkan dengan dense optical flow. Metode Lucas Kanade yang diterapkan pada sparse optical flow, terbukti lebih cepat dan lebih baik dalam mendeskripsikan fitur gerak dari api.

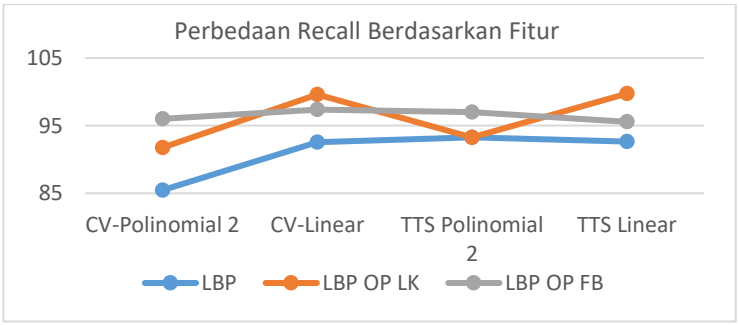
Uji coba 3 dan 6 menunjukkan bahwa penggunaan ekstraksi fitur optical flow sebagai tambahan dari ekstraksi fitur LBP mempengaruhi presentase keberhasilan sistem dalam mendeteksi api. Dengan menggunakan tambahan ektraksi fitur optical flow,

yaitu optical flow dengan metode Lucas Kanade tingkat keberhasilan meningkat sebesar 6%.



Gambar 5.12 Grafik Perbedaan Nilai Akurasi Berdasarkan Fitur

Berdasarkan Gambar 5.13 Penambahan fitur optical flow meningkat performa sistem dalam hal recall. Hal ini berarti optical flow mampu meningkatkan kepekaan (sensitivity) terhadap adanya api.



Gambar 5.13 Grafik Perbedaan recall Berdasarkan Fitur

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas tentang kesimpulan yang didasari oleh hasil uji coba yang telah dilakukan pada bab sebelumnya. Kesimpulan nantinya sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut di masa depan.

6.1 Kesimpulan

Dalam pengerjaan Tugas Akhir ini melalui tahap perancangan aplikasi, implementasi metode, serta uji coba. Sehingga mendapatkan kesimpulan sebagai berikut:

1. Deteksi warna api pada piksel frame video dilakukan dengan cara mencari rata-rata nilai piksel pada channel red, green, dan blue dari 6 gambar api, kemudian nilai rata rata dari piksel tersebut diubah kedalam format hue, saturation, value untuk dilakukan masking dengan threshold sebesar hue sebesar 20, saturation sebesar 100, dan value sebesar 70.
2. Deteksi tekstur dari citra yang telah disegmentasi pada area api dilakukan dengan menggunakan metode LBP pada citra grayscale frame. Metode ini membandingkan nilai piksel tersebut dengan 8 ketetanggaan, untuk mendapatkan nilai grayscale baru dengan range 0-255.
3. Deteksi gerak api dilakukan dengan cara menentukan titik titik yang dievaluasi pada area api, kemudian menggunakan metode optical flow, ditentukan perpindahan titik berikutnya dari titik awal. perpindahan dihitung menggunakan *eucledian distance*.
4. Fitur yang didapatkan dari proses deteksi tekstur berupa 256 fitur LBP dan 1 fitur optical flow dijadikan input dalam klasifikasi SVM. Pada percobaan ini, metode

klasifikasi SVM terbaik diperoleh menggunakan kernel linear.

5. Penggunaan ekstraksi fitur optical flow meningkatkan tingkat keberhasilan dibandingkan dengan hanya menggunakan ekstraksi fitur LBP pada sistem pendeteksi api, terutama pada recall. Rata rata recall naik sebesar 6%
6. Optical flow yang baik digunakan dalam sistem pendeteksi api ini adalah sparse optical flow, menggunakan metode Lucas Kanade
7. Penggunaan kernel pada klasifikasi mempengaruhi keberhasilan sistem dalam mendeteksi api. Kernel terbaik pada scenario uji coba 5 adalah kernel linear
8. Nilai akurasi terbaik diperoleh pada model dengan ekstraksi fitur LBP + Optical Flow Lucas Kanade menggunakan metode SVM dengan kernel Linear yaitu 95.96%, dengan presisi 93,76% dan recall sebesar 99,73%.
9. Metode pemisahan data dalam pembuatan model dalam hal ini metode *cross validation* dan *train test split*, tidak memiliki perbedaan pengaruh yang besar pada keberhasilan sistem dalam mendeteksi api.

6.2 Saran

Saran yang diberikan untuk pengembangan sistem deteksi api berbasis data video dengan metode optical flow dan support vector machines, yaitu:

1. Perlu dilakukan uji coba lebih lanjut pada parameter SVM yang lain, serta jumlah fold yang lain dalam pembuatan model.
2. Perlu dilakukan uji coba lebih lanjut mengenai penggunaan jenis LBP yang lain dan penggunaan metode optical flow lain.

Daftar Pustaka

[1] WorlBank, "Indonesia Sustainable Landscape Knowledge Note," *The Cost of Fire : An Economic Analysis of Indonesia's 2015 Fire Crisis*, p. 1, February 2016.

[2] H. A. Muzakkiy, "Deteksi Api Berbasis Sensor Visual menggunakan Metode Support Vector Machine," Institut Teknologi Sepuluh Nopember, Surabaya, 2016.

[3] Y. W. A. Wu, J. Zhang, M. Zhao, W. Li and N. Dong, "Fire Smoke Detection Based on Texture Features and Optical Flow Vector of Contour," in *World Congress on Intelligent Control and Automation (WCICA)*, Guilin, China, 2016.

[4] K. Avgerinakis, P. Giannakeris, A. Briassouli, A. Karakostas, S. Vrochidis and I. Kompatsiaris , "LBP-flow and Hybrid encoding for real time water and fire classification," *IEEE International Conference on computer Vison Workshop*, vol. 56, pp. 412-418, 2017.

[5] opencv, "Image Pyramid Opencv," 18 04 2018. [Online]. Available: http://docs.opencv.org/2.4/doc/tutorials/imgproc/pyram ids/pyramids.html.

[6] R. C. Gonzales, R. E. Woods and S. E. Eddins, Digital Image Processing using Matlab, Upper Saddle River, NJ, USA: Prentice-Hall, Inc, 2003.

[7] Z. Guo, L. Zhang and D. Zhang, "A completed modeling of local binary pattern operator for texture classification," *IEEE Trans. Image Process*, pp. 1657-1663, 2010.

[8] Opencv Org, "Optical Flow Python Tutorials OpenCv Documentation," 18 04 2018. [Online]. Available: https://docs.opencv.org/3.3.1/d7/d8b/tutorial_py_lucas_kanade.html.

[9] G. Farneback, "Two-frame motion estimation based on polynomial expansion," *Image analysis*, pp. 363-370, 2003.

[10] Surtoyo, Mulyanto Edi, Suhartono, Vincent, Nurhayati and Wijanarto, Pengolahan Citra Digital, Yogyakarta: Andi Offset, 2007.

[11] "About Python," Python, [Online]. Available: https://www.python.org/about/. [Accessed 2018 May 31].

[12] "OpenCV," [Online]. Available: https://opencv.org/. [Accessed 31 May 2018].

[13] "NumPy," NumPy, [Online]. Available: http://www.numpy.org/. [Accessed 31 May 2018].

[14] "Scikit-learn," Scikit-learn, [Online]. Available: http://scikit-learn.org/stable/index.html. [Accessed 31 May 2018].

[15] "Matplotlib," Matplotlib, [Online]. Available: https://matplotlib.org/index.html. [Accessed 31 May 2018].

[16] "Transformation matrix," Wikiwand, [Online]. Available: http://www.wikiwand.com/en/Transformation_matrix. [Accessed 6 May 2018].

[17] WordlBank, "The Cost of Fire: An Economic Analysis of Indonesia's 2015 Fire Crisis," *Indonesia Suistanable Landscape Knowledge Note*, p. 1, 1 February 2016.

[18] B. Ko, J. Y. Nam and K.-H. Cheong, "Early fire detection algorithm based on irregular patterns of flames and," *Fire Safety Journal*, pp. 262-270, 2007.

LAMPIRAN

L.1 Hasil Klasifikasi dengan Model Cross Validation Kernel Polynomial, Ekstraksi Fitur LBP+Optical Flow Lucas Kanade

No	File Video	True Positive dan True Negative (%)	False Positive (%)	False Negative (%)
1	Api11_ID_1.mp4	100.00	0.00	0.00
2	Api13_ID_1.mp4	100.00	0.00	0.00
3	Api14_ID_1.mp4	100.00	0.00	0.00
4	Api16_ID_0.mp4	100.00	0.00	0.00
5	Api17_ID_0.mp4	100.00	0.00	0.00
6	Api22_ID_0.mp4	100.00	0.00	0.00
7	Api23_ID_0.mp4	100.00	0.00	0.00
8	Api24_ID_0.mp4	100.00	0.00	0.00
9	Api25_ID_0.mp4	100.00	0.00	0.00
10	Api26_ID_0.mp4	100.00	0.00	0.00
11	Api27_ID_0.mp4	100.00	0.00	0.00
12	Api28_ID_0.mp4	100.00	0.00	0.00
13	Api2_ID_1.mp4	74.70	0.00	25.30
14	Api45_ID_1.mp4	97.31	0.00	2.69
15	ApiTest13_ID_0.mp4	90.00	10.00	0.00
16	ApiTest26_ID_1.mp4	99.43	0.00	0.57
17	ApiTest28_ID_0.mp4	54.12	45.88	0.00
18	ApiTest33_ID_1.mp4	97.74	0.00	2.26
19	ApiTest36_ID_1.mp4	100.00	0.00	0.00
20	ApiTest37_ID_1.mp4	99.31	0.00	0.69
21	Apitest38_ID_1.mp4	100.00	0.00	0.00
22	ApiTest3_ID_1.mp4	100.00	0.00	0.00

No	File Video	True Positive dan True Negative (%)	False Positive (%)	False Negative (%)
23	ApiTest40_ID_1.mp4	100.00	0.00	0.00
24	ApiTest_ID_1.mp4	16.94	0.00	83.06
25	Api_ID_1.mp4	100.00	0.00	0.00

L.2 Hasil Klasifikasi dengan Model Cross Validation Kernel Linear, Ekstraksi Fitur LBP+Optical Flow Lucas Kanade

No	File Video	True Positive dan True Negative (%)	False Positive (%)	False Negative (%)
1	Api11_ID_1.mp4	100.00	0.00	0.00
2	Api13_ID_1.mp4	100.00	0.00	0.00
3	Api14_ID_1.mp4	99.07	0.00	0.93
4	Api16_ID_0.mp4	100.00	0.00	0.00
5	Api17_ID_0.mp4	100.00	0.00	0.00
6	Api22_ID_0.mp4	100.00	0.00	0.00
7	Api23_ID_0.mp4	100.00	0.00	0.00
8	Api24_ID_0.mp4	100.00	0.00	0.00
9	Api25_ID_0.mp4	100.00	0.00	0.00
10	Api26_ID_0.mp4	100.00	0.00	0.00
11	Api27_ID_0.mp4	100.00	0.00	0.00
12	Api28_ID_0.mp4	97.73	2.27	0.00
13	Api2_ID_1.mp4	100.00	0.00	0.00
14	Api45_ID_1.mp4	99.10	0.00	0.90
15	ApiTest13_ID_0.mp4	92.20	7.80	0.00
16	ApiTest26_ID_1.mp4	99.43	0.00	0.57
17	ApiTest28_ID_0.mp4	50.26	49.74	0.00
18	ApiTest33_ID_1.mp4	99.06	0.00	0.94

No	File Video	True Positive dan True Negative (%)	False Positive (%)	False Negative (%)
19	ApiTest36_ID_1.mp4	100.00	0.00	0.00
20	ApiTest37_ID_1.mp4	99.31	0.00	0.69
21	Apitest38_ID_1.mp4	100.00	0.00	0.00
22	ApiTest3_ID_1.mp4	100.00	0.00	0.00
23	ApiTest40_ID_1.mp4	99.24	0.00	0.76
24	ApiTest_ID_1.mp4	100.00	0.00	0.00
25	Api_ID_1.mp4	100.00	0.00	0.00
	Rata-Rata	**97.42**	**2.39**	**0.19**

L.3 Hasil Klasifikasi dengan Model Train Test Split Kernel Polynomial, Ekstraksi Fitur LBP+Optical Flow Lucas Kanade

No	File Video	True Positive dan True Negative (%)	False Positive (%)	False Negative (%)
1	Api11_ID_1.mp4	100.00	0.00	0.00
2	Api13_ID_1.mp4	100.00	0.00	0.00
3	Api14_ID_1.mp4	100.00	0.00	0.00
4	Api16_ID_0.mp4	100.00	0.00	0.00
5	Api17_ID_0.mp4	100.00	0.00	0.00
6	Api22_ID_0.mp4	100.00	0.00	0.00
7	Api23_ID_0.mp4	100.00	0.00	0.00
8	Api24_ID_0.mp4	100.00	0.00	0.00
9	Api25_ID_0.mp4	100.00	0.00	0.00
10	Api26_ID_0.mp4	100.00	0.00	0.00
11	Api27_ID_0.mp4	100.00	0.00	0.00
12	Api28_ID_0.mp4	100.00	0.00	0.00
13	Api2_ID_1.mp4	100.00	0.00	0.00

No	File Video	True Positive dan True Negative (%)	False Positive (%)	False Negative (%)
14	Api45_ID_1.mp4	96.71	0.00	3.29
15	ApiTest13_ID_0.mp4	88.05	11.95	0.00
16	ApiTest26_ID_1.mp4	97.99	0.00	2.01
17	ApiTest28_ID_0.mp4	56.04	43.96	0.00
18	ApiTest33_ID_1.mp4	95.67	0.00	4.33
19	ApiTest36_ID_1.mp4	100.00	0.00	0.00
20	ApiTest37_ID_1.mp4	99.08	0.00	0.92
21	Apitest38_ID_1.mp4	99.32	0.00	0.68
22	ApiTest3_ID_1.mp4	100.00	0.00	0.00
23	ApiTest40_ID_1.mp4	100.00	0.00	0.00
24	ApiTest_ID_1.mp4	15.83	0.00	84.17
25	Api_ID_1.mp4	100.00	0.00	0.00

L.4 Hasil Klasifikasi dengan Model Train Test Split Kernel Linear, Ekstraksi Fitur LBP+Optical Flow Lucas Kanade

No	File Video	True Positive dan True Negative (%)	False Positive (%)	False Negative (%)
1	Api11_ID_1.mp4	100.00	0.00	0.00
2	Api13_ID_1.mp4	100.00	0.00	0.00
3	Api14_ID_1.mp4	99.07	0.00	0.93
4	Api16_ID_0.mp4	100.00	0.00	0.00
5	Api17_ID_0.mp4	100.00	0.00	0.00
6	Api22_ID_0.mp4	100.00	0.00	0.00
7	Api23_ID_0.mp4	100.00	0.00	0.00
8	Api24_ID_0.mp4	100.00	0.00	0.00
9	Api25_ID_0.mp4	100.00	0.00	0.00

No	File Video	True Positive dan True Negative (%)	False Positive (%)	False Negative (%)
10	Api26_ID_0.mp4	100.00	0.00	0.00
11	Api27_ID_0.mp4	100.00	0.00	0.00
12	Api28_ID_0.mp4	92.05	7.95	0.00
13	Api2_ID_1.mp4	100.00	0.00	0.00
14	Api45_ID_1.mp4	98.50	0.00	1.50
15	ApiTest13_ID_0.mp4	92.44	7.56	0.00
16	ApiTest26_ID_1.mp4	100.00	0.00	0.00
17	ApiTest28_ID_0.mp4	46.06	53.94	0.00
18	ApiTest33_ID_1.mp4	99.06	0.00	0.94
19	ApiTest36_ID_1.mp4	100.00	0.00	0.00
20	ApiTest37_ID_1.mp4	100.00	0.00	0.00
21	Apitest38_ID_1.mp4	100.00	0.00	0.00
22	ApiTest3_ID_1.mp4	100.00	0.00	0.00
23	ApiTest40_ID_1.mp4	100.00	0.00	0.00
24	ApiTest_ID_1.mp4	100.00	0.00	0.00
25	Api_ID_1.mp4	100.00	0.00	0.00

L.5 Hasil Klasifikasi dengan Model Cross Validation Kernel Polynomial, Ekstraksi Fitur LBP+Optical Flow Farne Back

No	File Video	True Positive dan True Negative (%)	False Positive (%)	False Negative (%)
1	Api11_ID_1.mp4	100.00	0.00	0.00
2	Api13_ID_1.mp4	100.00	0.00	0.00
3	Api14_ID_1.mp4	100.00	0.00	0.00
4	Api16_ID_0.mp4	100.00	0.00	0.00
5	Api17_ID_0.mp4	22.34	77.66	0.00

No	File Video	True Positive dan True Negative (%)	False Positive (%)	False Negative (%)
6	Api22_ID_0.mp4	47.96	52.04	0.00
7	Api23_ID_0.mp4	47.91	52.09	0.00
8	Api24_ID_0.mp4	100.00	0.00	0.00
9	Api25_ID_0.mp4	63.94	36.06	0.00
10	Api26_ID_0.mp4	4.83	95.17	0.00
11	Api27_ID_0.mp4	100.00	0.00	0.00
12	Api28_ID_0.mp4	0.00	100.00	0.00
13	Api2_ID_1.mp4	100.00	0.00	0.00
14	Api45_ID_1.mp4	83.53	0.00	16.47
15	ApiTest13_ID_0.mp4	96.59	3.41	0.00
16	ApiTest26_ID_1.mp4	97.99	0.00	2.01
17	ApiTest28_ID_0.mp4	79.68	20.32	0.00
18	ApiTest33_ID_1.mp4	94.54	0.00	5.46
19	ApiTest36_ID_1.mp4	100.00	0.00	0.00
20	ApiTest37_ID_1.mp4	99.77	0.00	0.23
21	Apitest38_ID_1.mp4	97.96	0.00	2.04
22	ApiTest3_ID_1.mp4	100.00	0.00	0.00
23	ApiTest40_ID_1.mp4	98.99	0.00	1.01
24	ApiTest_ID_1.mp4	71.11	0.00	28.89
25	Api_ID_1.mp4	100.00	0.00	0.00

L.6 Hasil Klasifikasi dengan Model Cross Validation Kernel Linear, Ekstraksi Fitur LBP+Optical Flow Farne Back

No	File Video	True Positive dan True Negative (%)	False Positive (%)	False Negative (%)
1	Api11_ID_1.mp4	100.00	0.00	0.00

No	File Video	True Positive dan True Negative (%)	False Positive (%)	False Negative (%)
2	Api13_ID_1.mp4	100.00	0.00	0.00
3	Api14_ID_1.mp4	100.00	0.00	0.00
4	Api16_ID_0.mp4	100.00	0.00	0.00
5	Api17_ID_0.mp4	23.59	76.41	0.00
6	Api22_ID_0.mp4	48.98	51.02	0.00
7	Api23_ID_0.mp4	45.12	54.88	0.00
8	Api24_ID_0.mp4	100.00	0.00	0.00
9	Api25_ID_0.mp4	56.27	43.73	0.00
10	Api26_ID_0.mp4	4.53	95.47	0.00
11	Api27_ID_0.mp4	100.00	0.00	0.00
12	Api28_ID_0.mp4	81.82	18.18	0.00
13	Api2_ID_1.mp4	100.00	0.00	0.00
14	Api45_ID_1.mp4	95.21	0.00	4.79
15	ApiTest13_ID_0.mp4	96.59	3.41	0.00
16	ApiTest26_ID_1.mp4	98.85	0.00	1.15
17	ApiTest28_ID_0.mp4	68.48	31.52	0.00
18	ApiTest33_ID_1.mp4	95.29	0.00	4.71
19	ApiTest36_ID_1.mp4	96.23	0.00	3.77
20	ApiTest37_ID_1.mp4	98.16	0.00	1.84
21	Apitest38_ID_1.mp4	97.29	0.00	2.71
22	ApiTest3_ID_1.mp4	96.23	0.00	3.77
23	ApiTest40_ID_1.mp4	99.50	0.00	0.50
24	ApiTest_ID_1.mp4	81.67	0.00	18.33
25	Api_ID_1.mp4	100.00	0.00	0.00

L.7 Hasil Klasifikasi dengan Model Train Test Split Kernel Polynomial, Ekstraksi Fitur LBP+Optical Flow Farne Back

No	File Video	True Positive dan True Negative (%)	False Positive (%)	False Negative (%)
1	Api11_ID_1.mp4	100.00	0.00	0.00
2	Api13_ID_1.mp4	100.00	0.00	0.00
3	Api14_ID_1.mp4	100.00	0.00	0.00
4	Api16_ID_0.mp4	100.00	0.00	0.00
5	Api17_ID_0.mp4	22.34	77.66	0.00
6	Api22_ID_0.mp4	51.36	48.64	0.00
7	Api23_ID_0.mp4	45.58	54.42	0.00
8	Api24_ID_0.mp4	100.00	0.00	0.00
9	Api25_ID_0.mp4	57.54	42.46	0.00
10	Api26_ID_0.mp4	4.83	95.17	0.00
11	Api27_ID_0.mp4	100.00	0.00	0.00
12	Api28_ID_0.mp4	35.23	64.77	0.00
13	Api2_ID_1.mp4	100.00	0.00	0.00
14	Api45_ID_1.mp4	97.31	0.00	2.69
15	ApiTest13_ID_0.mp4	97.56	2.44	0.00
16	ApiTest26_ID_1.mp4	97.99	0.00	2.01
17	ApiTest28_ID_0.mp4	73.91	26.09	0.00
18	ApiTest33_ID_1.mp4	95.67	0.00	4.33
19	ApiTest36_ID_1.mp4	100.00	0.00	0.00
20	ApiTest37_ID_1.mp4	99.54	0.00	0.46
21	Apitest38_ID_1.mp4	97.96	0.00	2.04
22	ApiTest3_ID_1.mp4	100.00	0.00	0.00
23	ApiTest40_ID_1.mp4	98.49	0.00	1.51

No	File Video	True Positive dan True Negative (%)	False Positive (%)	False Negative (%)
24	ApiTest_ID_1.mp4	72.22	0.00	27.78
25	Api_ID_1.mp4	100.00	0.00	0.00

L.8 Hasil Klasifikasi dengan Model Train Test Split Kernel Linear, Ekstraksi Fitur LBP+Optical Flow Farne Back

No	File Video	True Positive dan True Negative (%)	False Positive (%)	False Negative (%)
1	Api11_ID_1.mp4	100.00	0.00	0.00
2	Api13_ID_1.mp4	100.00	0.00	0.00
3	Api14_ID_1.mp4	100.00	0.00	0.00
4	Api16_ID_0.mp4	100.00	0.00	0.00
5	Api17_ID_0.mp4	22.96	77.04	0.00
6	Api22_ID_0.mp4	47.96	52.04	0.00
7	Api23_ID_0.mp4	47.91	52.09	0.00
8	Api24_ID_0.mp4	100.00	0.00	0.00
9	Api25_ID_0.mp4	60.61	39.39	0.00
10	Api26_ID_0.mp4	4.23	95.77	0.00
11	Api27_ID_0.mp4	100.00	0.00	0.00
12	Api28_ID_0.mp4	86.36	13.64	0.00
13	Api2_ID_1.mp4	100.00	0.00	0.00
14	Api45_ID_1.mp4	87.43	0.00	12.57
15	ApiTest13_ID_0.mp4	97.07	2.93	0.00
16	ApiTest26_ID_1.mp4	99.71	0.00	0.29
17	ApiTest28_ID_0.mp4	67.25	32.75	0.00
18	ApiTest33_ID_1.mp4	92.09	0.00	7.91
19	ApiTest36_ID_1.mp4	96.23	0.00	3.77

No	File Video	True Positive dan True Negative (%)	False Positive (%)	False Negative (%)
20	ApiTest37_ID_1.mp4	94.48	0.00	5.52
21	Apitest38_ID_1.mp4	95.25	0.00	4.75
22	ApiTest3_ID_1.mp4	96.23	0.00	3.77
23	ApiTest40_ID_1.mp4	99.24	0.00	0.76
24	ApiTest_ID_1.mp4	73.89	0.00	26.11
25	Api_ID_1.mp4	100.00	0.00	0.00

L.9 Hasil Klasifikasi dengan Model Cross Validation Kernel Polynomial, Ekstraksi Fitur LBP

No	File Video	True Positive dan True Negative (%)	False Positive (%)	False Negative (%)
1	Api11_ID_1.mp4	100.00	0.00	0.00
2	Api13_ID_1.mp4	100.00	0.00	0.00
3	Api14_ID_1.mp4	100.00	0.00	0.00
4	Api16_ID_0.mp4	100.00	0.00	0.00
5	Api17_ID_0.mp4	22.34	77.66	0.00
6	Api22_ID_0.mp4	100.00	0.00	0.00
7	Api23_ID_0.mp4	100.00	0.00	0.00
8	Api24_ID_0.mp4	100.00	0.00	0.00
9	Api25_ID_0.mp4	100.00	0.00	0.00
10	Api26_ID_0.mp4	100.00	0.00	0.00
11	Api27_ID_0.mp4	100.00	0.00	0.00
12	Api28_ID_0.mp4	56.82	43.18	0.00
13	Api2_ID_1.mp4	90.12	0.00	9.88
14	Api45_ID_1.mp4	35.03	0.00	64.97
15	ApiTest13_ID_0.mp4	97.80	2.20	0.00

No	File Video	True Positive dan True Negative (%)	False Positive (%)	False Negative (%)
16	ApiTest26_ID_1.mp4	97.13	0.00	2.87
17	ApiTest28_ID_0.mp4	95.97	4.03	0.00
18	ApiTest33_ID_1.mp4	93.60	0.00	6.40
19	ApiTest36_ID_1.mp4	94.34	0.00	5.66
20	ApiTest37_ID_1.mp4	98.85	0.00	1.15
21	Apitest38_ID_1.mp4	9.95	0.00	90.05
22	ApiTest3_ID_1.mp4	94.34	0.00	5.66
23	ApiTest40_ID_1.mp4	89.17	0.00	10.83
24	ApiTest_ID_1.mp4	98.89	0.00	1.11
25	Api_ID_1.mp4	100.00	0.00	0.00

L.10 Hasil Klasifikasi dengan Model Cross Validation Kernel Linear, Ekstraksi Fitur LBP

No	File Video	True Positive dan True Negative (%)	False Positive (%)	False Negative (%)
1	Api11_ID_1.mp4	100.00	0.00	0.00
2	Api13_ID_1.mp4	100.00	0.00	0.00
3	Api14_ID_1.mp4	100.00	0.00	0.00
4	Api16_ID_0.mp4	100.00	0.00	0.00
5	Api17_ID_0.mp4	22.96	77.04	0.00
6	Api22_ID_0.mp4	100.00	0.00	0.00
7	Api23_ID_0.mp4	100.00	0.00	0.00
8	Api24_ID_0.mp4	100.00	0.00	0.00
9	Api25_ID_0.mp4	100.00	0.00	0.00
10	Api26_ID_0.mp4	100.00	0.00	0.00
11	Api27_ID_0.mp4	100.00	0.00	0.00

No	File Video	True Positive dan True Negative (%)	False Positive (%)	False Negative (%)
12	Api28_ID_0.mp4	89.77	10.23	0.00
13	Api2_ID_1.mp4	100.00	0.00	0.00
14	Api45_ID_1.mp4	99.70	0.00	0.30
15	ApiTest13_ID_0.mp4	97.56	2.44	0.00
16	ApiTest26_ID_1.mp4	99.43	0.00	0.57
17	ApiTest28_ID_0.mp4	67.78	32.22	0.00
18	ApiTest33_ID_1.mp4	94.92	0.00	5.08
19	ApiTest36_ID_1.mp4	92.45	0.00	7.55
20	ApiTest37_ID_1.mp4	95.86	0.00	4.14
21	Apitest38_ID_1.mp4	99.10	0.00	0.90
22	ApiTest3_ID_1.mp4	92.45	0.00	7.55
23	ApiTest40_ID_1.mp4	98.49	0.00	1.51
24	ApiTest_ID_1.mp4	10.56	0.00	89.44
25	Api_ID_1.mp4	100.00	0.00	0.00

L.11 Hasil Klasifikasi dengan Model Train Test Split Kernel Polynomial, Ekstraksi Fitur LBP

No	File Video	True Positive dan True Negative (%)	False Positive (%)	False Negative (%)
1	Api11_ID_1.mp4	100.00	0.00	0.00
2	Api13_ID_1.mp4	100.00	0.00	0.00
3	Api14_ID_1.mp4	99.77	0.23	0.00
4	Api16_ID_0.mp4	100.00	0.00	0.00
5	Api17_ID_0.mp4	22.34	0.00	77.66
6	Api22_ID_0.mp4	100.00	0.00	0.00
7	Api23_ID_0.mp4	100.00	0.00	0.00

No	File Video	True Positive dan True Negative (%)	False Positive (%)	False Negative (%)
8	Api24_ID_0.mp4	100.00	0.00	0.00
9	Api25_ID_0.mp4	100.00	0.00	0.00
10	Api26_ID_0.mp4	100.00	0.00	0.00
11	Api27_ID_0.mp4	100.00	0.00	0.00
12	Api28_ID_0.mp4	57.95	0.00	42.05
13	Api2_ID_1.mp4	90.84	9.16	0.00
14	Api45_ID_1.mp4	35.33	64.67	0.00
15	ApiTest13_ID_0.mp4	98.05	0.00	1.95
16	ApiTest26_ID_1.mp4	97.13	2.87	0.00
17	ApiTest28_ID_0.mp4	89.49	0.00	10.51
18	ApiTest33_ID_1.mp4	93.22	6.78	0.00
19	ApiTest36_ID_1.mp4	100.00	0.00	0.00
20	ApiTest37_ID_1.mp4	97.70	2.30	0.00
21	Apitest38_ID_1.mp4	9.95	90.05	0.00
22	ApiTest3_ID_1.mp4	100.00	0.00	0.00
23	ApiTest40_ID_1.mp4	94.71	5.29	0.00
24	ApiTest_ID_1.mp4	100.00	0.00	0.00
25	Api_ID_1.mp4	100.00	0.00	0.00

L.12 Hasil Klasifikasi dengan Model Train Test Split Kernel Linear, Ekstraksi Fitur LBP

No	File Video	True Positive dan True Negative (%)	False Positive (%)	False Negative (%)
1	Api11_ID_1.mp4	100.00	0.00	0.00
2	Api13_ID_1.mp4	100.00	0.00	0.00
3	Api14_ID_1.mp4	100.00	0.00	0.00

No	File Video	True Positive dan True Negative (%)	False Positive (%)	False Negative (%)
4	Api16_ID_0.mp4	100.00	0.00	0.00
5	Api17_ID_0.mp4	22.34	77.66	0.00
6	Api22_ID_0.mp4	100.00	0.00	0.00
7	Api23_ID_0.mp4	100.00	0.00	0.00
8	Api24_ID_0.mp4	100.00	0.00	0.00
9	Api25_ID_0.mp4	100.00	0.00	0.00
10	Api26_ID_0.mp4	100.00	0.00	0.00
11	Api27_ID_0.mp4	100.00	0.00	0.00
12	Api28_ID_0.mp4	92.05	7.95	0.00
13	Api2_ID_1.mp4	100.00	0.00	0.00
14	Api45_ID_1.mp4	99.70	0.00	0.30
15	ApiTest13_ID_0.mp4	98.54	1.46	0.00
16	ApiTest26_ID_1.mp4	97.99	0.00	2.01
17	ApiTest28_ID_0.mp4	66.90	33.10	0.00
18	ApiTest33_ID_1.mp4	96.42	0.00	3.58
19	ApiTest36_ID_1.mp4	98.11	0.00	1.89
20	ApiTest37_ID_1.mp4	96.55	0.00	3.45
21	Apitest38_ID_1.mp4	98.87	0.00	1.13
22	ApiTest3_ID_1.mp4	98.11	0.00	1.89
23	ApiTest40_ID_1.mp4	98.49	0.00	1.51
24	ApiTest_ID_1.mp4	8.89	0.00	91.11
25	Api_ID_1.mp4	100.00	0.00	0.00

L.13 Hasil Klasifikasi dengan Model Cross Validation Kernel Polynomial, Ekstraksi Fitur LBP

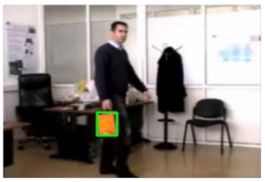
Api11_ID_1

Api_13_ID_3

Api14_ID_1

Api16_ID_0

Api17_ID_0



Api22_ID_0

Api23_ID_0

Api24_ID_0

Api25_ID_0

Api26_ID_0

Api27_ID_0

Api28_ID_0

Api2_ID_1

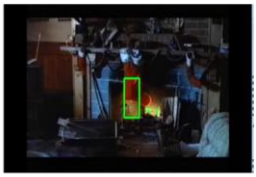
Api45_ID_1

ApiTest13_ID_0

ApiTest26_ID_1

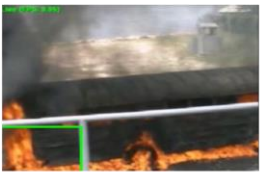
ApiTest28_ID_0

ApiTest33_ID_1



ApiTest36_ID_1

ApiTest37_ID_1



Apitest38_ID_1

ApiTest3_ID_1

ApiTest40_ID_1

ApiTest_ID_1

Api_ID_1

BIODATA PENULIS

Sirria Panah Alam, lahir di Blitar pada tanggal 25 Oktober 1996. Penulis menempuh pendidikan mulai dari TK Al Hidayah Bence II (2001-2003), SD Negeri Sananwetan 3 (2003-2009), SMP Negeri 1 Blitar (2009-2012), SMA Negeri 1 Blitar (2012-2015), dan sekarang sedang menjalani pendidikan S1 Informatika di ITS. Penulis aktif dalam organisasi dan kepanitiaan Himpunan Mahasiswa Teknik Computer (HMTC) dan Schematics. Diantaranya adalah menjadi staff departemen teknologi HMTC ITS 2016-2017 serta panitia dalam National Logic Competition (NLC) Schematics 2016-2017. Penulis juga aktif dalam kegiatan pengembangan dan inovasi teknologi. Diantaranya penulis pernah berpartisipasi dalam lomba Gemastik sebagai finalis dan juara. Penulis juga pernah menjadi salah satu anggota tim Robot Terbang VTOL ITS sebagai programmer. Komunikasi dengan penulis dapat melalui telepon: +6282335619807 dan *email*: **sirriapalam@gmail.com**.